



UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

Proyecto de fin de Grado en Ingeniería Informática

Hacia la superresolución espectral en líneas de absorción solares a partir de imágenes pseudomonocromáticas

Fernando H. Nasser-Eddine López

Dirigido por: Pablo Rodríguez Gil

Codirigido por: Luis Manuel Sarro Baro

Curso 2023/2024, convocatoria junio



Hacia la superresolución espectral en líneas de absorción solares a partir de imágenes pseudomonocromáticas

Proyecto de fin de Grado en Ingeniería Informática
de modalidad externa

Realizado por: Fernando H. Nasser-Eddine López

Dirigido por: Pablo Rodríguez Gil

Codirigido por: Luis Manuel Sarro Baro

Fecha de lectura y defensa: 26 de junio de 2024

Agradecimientos

A Andrés Asensio, que fue el verdadero artista detrás de la propuesta.

A Luis Sarro, por su apoyo y comprensión, en lo académico y en lo personal.

A mi amigo Pablo Rodríguez por su excepcional mentoría y su infinita paciencia.

A Laura y Jimena por iluminar cada minuto de mi vida.

Resumen

En este proyecto de fin de grado (PFG), exploramos la viabilidad de aplicar técnicas modernas de aprendizaje profundo para abordar el desafío de la superresolución espectral. Nos centramos en un conjunto de 21 imágenes de alta resolución que representan diferentes longitudes de onda en la línea de absorción de hierro neutro(Fe I λ 6302) en 6302 Å de la fotosfera del Sol.

Partiendo del estado del arte en técnicas de regresión de imágenes, tales como *SINusoidal REpresentation Networks* (SIREN) y *Fourier Features*, nuestro primer objetivo fue seleccionar la arquitectura que ofreciera la mejor aproximación de nuestras imágenes solares utilizando la métrica de la relación señal-ruido (PSNR). Tras lograr resultados comparables con estos estudios, dimos un paso adicional y extendimos la funcionalidad de la arquitectura al espacio completo de nuestro conjunto de datos incluyendo la longitud de onda para cada píxel de nuestras imágenes, es decir, $I(x, y, \lambda)$. Esto nos permitió generar imágenes en longitudes de onda intermedias a las 21 originales usando un *neural field*.

En este PFG no solo confirmamos la eficacia de las técnicas de aprendizaje profundo en la aproximación de imágenes pseudomonocromáticas de alta calidad, sino que también extendemos estos métodos al dominio de la superresolución espectral. Al hacerlo, contribuimos a futuras investigaciones que podrían mejorar la forma en que capturamos y procesamos datos espectrales para su uso en multitud de campos desde la astrofísica a la biomedicina.

Palabras clave

Física solar, procesamiento de imágenes, superresolución espectral, inteligencia artificial, visión artificial, aprendizaje profundo, redes neuronales, campos neurales, reconstrucción de imágenes, regresión de imágenes, redes implícitas sinusoidales (SIREN), características de Fourier.



Towards spectral super-resolution in solar absorption lines from pseudomonochromatic images

Proyecto de fin de Grado en Ingeniería Informática
de modalidad externa

Abstract

In this Bachelor’s Degree Final Project (PFG), we explored the feasibility of applying modern deep learning techniques to address the challenge of spectral super-resolution. We focused on a set of 21 high-resolution images representing different wavelengths in the absorption line of neutral iron ($\text{Fe I } \lambda 6302$) at 6302 \AA of the Sun’s photosphere.

Starting from the state-of-the-art in image regression techniques, such as *SINusoidal REpresentation Networks* (SIREN) and *Fourier Features*, our first objective was to select the architecture that offered the best approximation of our solar images using the signal-to-noise ratio (PSNR) metric. After achieving comparable results with these studies, we took an additional step and extended the functionality of the architecture to the complete space of our dataset including the wavelength for each pixel of our images, i.e., $I(x, y, \lambda)$. This allowed us to generate images at intermediate wavelengths to the original 21 using a *neural field*.

In this PFG, we not only confirmed the efficacy of deep learning techniques in the approximation of high-quality pseudomonochromatic images but also extended these methods to the domain of spectral super-resolution. By doing so, we contribute to future research that could improve the way we capture and process spectral data for use in a multitude of fields from astrophysics to biomedicine.

X

Keywords

Solar physics, image processing, spectral super-resolution, artificial intelligence, computer vision, deep learning, neural networks, neural fields, image reconstruction, image regression, sinusoidal representation networks (SIREN), Fourier features.

Índice

Índice de tablas	XV
Índice de figuras	XX
1. Introducción	1
1.1. Espectroscopía tridimensional del Sol	3
1.2. Motivación y objetivos	4
2. Campos neuronales	9
2.1. Campos	9
2.2. Redes neuronales	9
2.2.1. Entrenamiento de una red neuronal	12
2.3. Campos neurales (<i>neural fields</i>)	17
3. Regresión de imágenes bidimensionales	19
3.1. Conceptos previos	19
3.1.1. La inicialización de los parámetros	19
3.1.2. El ajuste de los hiperparámetros	21
3.1.3. Los algoritmos de optimización	22
3.1.4. La presentación de los datos de entrada	23
3.2. La reconstrucción de imágenes bidimensionales	23
3.2.1. Redes implícitas sinusoidales (SIREN)	23
3.2.2. Características de Fourier (<i>Fourier features</i>)	25
3.3. Experimentos con las imágenes solares	26
3.3.1. Regresión de imágenes bidimensionales con SIREN	27
3.3.2. Regresión de imágenes bidimensionales con características de Fourier (<i>Fourier features</i>)	30

3.4. Discusión de los resultados	33
4. Hacia la superresolución espectral	37
4.1. El conjunto de datos	37
4.2. Reconstrucción del cubo de imágenes solares	39
4.2.1. Experimentos con SIREN	40
4.2.2. Experimentos con <i>Fourier features</i>	41
4.3. Hacia la Superresolución Espectral	46
4.4. Conclusiones	49
Bibliografía	55
Lista de abreviaturas y acrónimos	59
A. Análisis del rendimiento de los modelos	61
A.1. Infraestructura utilizada	61
A.2. Coste Temporal	62
A.3. Coste Espacial	63
A.4. Rendimiento obtenido con <i>pytorch profiler</i>	64

Índice de tablas

3.1.	Resumen de los parámetros utilizados en el experimento de regresión de una imagen bidimensional usando SIREN	28
3.2.	Valores de la PSNR para los datos de entrenamiento y prueba en función de diferentes valores de ω_0 en el experimento de regresión de imágenes bidimensionales con SIREN.	29
3.3.	Resumen de los parámetros utilizados en el experimento de regresión de una imagen bidimensional usando características de Fourier (<i>Fourier features</i>).	32
3.4.	Resultados de la relación señal-ruido (PSNR) en la tarea de regresión de la imagen bidimensional para los conjuntos de datos de entrenamiento y prueba, con los diferentes mapeos de las características de Fourier.	32
4.1.	Valores de la PSNR para los datos de entrenamiento y prueba en función de diferentes valores de ω_0 en el experimento de regresión de las 21 imágenes psudomonocromáticas con SIREN.	41
4.2.	Resultados de la relación señal-ruido (PSNR) en el ajuste de los parámetros k y σ para los conjuntos de datos de prueba utilizando los valores de $k = 128$, $k = 256$, $k = 512$ y $\sigma = 0.8$, $\sigma = 10$, y $\sigma = 25$ durante 20 épocas	43
4.3.	Resumen de los parámetros utilizados en el experimento de regresión del cubo de datos tridimensional usando un <i>neural field</i> con características de Fourier (<i>Fourier features</i>).	45
4.4.	Resultados de la relación señal-ruido (PSNR) en la tarea de regresión de la totalidad de las imágenes para los conjuntos de datos de entrenamiento utilizando, por un lado, los valores de $k = 128$, $k = 512$ y por otro un escalar $\sigma = 0.8$ y un vector $\sigma = [10, 10, 0.8]$	45
A.1.	Comparativa de infraestructuras disponibles en Google Colab, incluyendo los costes operativos y las capacidades de procesamiento.	62
A.2.	Comparación de parámetros entre Fourier features y SIREN con GPUs T4 y A100	65
A.3.	Comparación de uso de memoria entre Fourier features y SIREN con GPUs T4 y A100	66

Índice de figuras

1.1.	Líneas espectrales de absorción en el espectro solar.	1
1.2.	Variación de la temperatura en la fotosfera y formación de una línea de absorción.	2
1.3.	De arriba a abajo, variación de la cantidad de absorción a medida que ascendemos en la fotosfera solar.	2
1.4.	Esquema de la formación de las líneas H y K de absorción del Ca II. Los fotones con energías cercanas a una transición electrónica tienen mayor probabilidad de ser absorbidos antes de escapar, por lo que los que llegan a la Tierra provienen de niveles más altos y fríos de la fotosfera solar.	3
1.5.	Imagen sintética de una región del Sol donde se observa la granulación solar (izquierda). Para cada uno de sus píxeles también se dispone de un espectro (derecha). Como ejemplo se muestra el espectro, que contiene dos líneas de absorción, de un píxel brillante (amarillo) y el de uno oscuro (azul).	4
1.6.	Configuración de un interferómetro de Fabry-Perot. Los rayos que provienen de un punto de la fuente P emergen paralelos tras la lente convergente L_1 . Estos rayos A , B , C entran con el mismo ángulo de inclinación θ en el espacio entre los dos reflectores parciales separados por una distancia d . A su salida, los rayos (R_1 , R_2 , ...) se dirigen al plano focal de la lente L_2 y convergen en el mismo punto Q de la pantalla o detector.	4
1.7.	Ejemplo ilustrativo de un cubo de seis imágenes de una región de la fotosfera solar en seis longitudes de onda distintas.	5
1.8.	Cada línea vertical indica alguna de las 21 bandas espectrales pseudomonocromáticas en las que se escanea la línea de absorción de Fe I $\lambda 6302$	6
1.9.	Tres de las 21 imágenes del escaneo de la línea de absorción de Fe I $\lambda 6302$ en la misma región del Sol. Nótense las diferentes intensidades en distintas partes de la línea.	6
2.1.	Modelo neuronal de McCulloch-Pitts en el que se incluye el conjunto de datos de entrada x_1, x_2, \dots, x_n y sus pesos asociados w_1, w_2, \dots, w_n . La letra a corresponde a la suma ponderada de las entradas más el sesgo b . La letra h representa la función de transferencia o activación que se aplica a la suma ponderada para obtener la salida z	10
2.2.	Funciones de activación h más comunes.	12
2.3.	Configuración simple de una red neuronal que cuenta con una capa de entrada, una capa oculta y una capa de salida.	13

2.4. Evolución de la función de pérdida $\mathcal{L} = \frac{1}{N} \sum_{i=1}^N (z_{\text{true},i} - z_{s,i})^2$	14
2.5. Representación de cómo se produce el flujo de datos y la retropropagación en una neurona basada en la regla de la cadena del cálculo diferencial.	16
3.1. Configuración de la red neuronal para el experimento con SIREN. La entrada está formada por las coordenadas espaciales de la imagen normalizadas al intervalo $[0, 1]$, consta de 3 capas ocultas de 256 neuronas y la capa de salida $\mathcal{I}(\mathbf{x}_l)$, que representa la intensidad del píxel para las coordenadas de entrada.	27
3.2. Evolución de la PSNR para los datos de entrenamiento (<i>train</i>) y prueba (<i>test</i>) con diferentes valores de ω a lo largo de las 2000 épocas (<i>epochs</i>).	29
3.3. Reconstrucción de una porción de 512×512 píxeles de una imagen del conjunto de datos utilizando la arquitectura SIREN. Las imágenes representan reconstrucciones con el conjunto de datos de prueba para diferentes valores de ω_0 y la imagen original (Ground truth, GT).	30
3.4. Configuración de la red neuronal para el experimento de características de Fourier. La entrada está formada por las coordenadas espaciales de la imagen $\mathbf{x}_l = (x_i, y_j)$ normalizadas a la rejilla unidad en el intervalo $[0, 1]$, que posteriormente se mapean para obtener el vector de características de Fourier (Ec. 3.10) con k neuronas (en este caso 256). La red tiene 3 capas ocultas con 256 neuronas cada una. Por último, la capa de salida representa la intensidad del píxel aprendida.	31
3.5. Evolución de la relación señal-ruido (PSNR) a lo largo de las 2000 épocas de entrenamiento de la red para los conjuntos de entrenamiento (<i>train</i>) y prueba (<i>test</i>). Cada una de las líneas corresponde al mapeo utilizado en el entrenamiento: Sin mapeo (<i>none</i>): $\gamma(\mathbf{v}) = \mathbf{v}$, mapeo básico (<i>basic</i>): $\gamma(\mathbf{v}) = [\cos(2\pi\mathbf{v}), \sin(2\pi\mathbf{v})]^T$ y mapeo de características de Fourier gaussiano: $\gamma(\mathbf{v}) = [\cos(2\pi\mathbf{B}\mathbf{v}), \sin(2\pi\mathbf{B}\mathbf{v})]^T$, con $\sigma = 1$, $\sigma = 10$ y $\sigma = 100$	33
3.6. Resultado de la regresión de una porción de 512×512 píxeles de una las imágenes de nuestro conjunto de datos usando características de Fourier. Se muestran las imágenes reconstruidas usando los datos de prueba y diferente mapeos: Sin mapeo (<i>none</i>): $\gamma(\mathbf{v}) = \mathbf{v}$, mapeo básico (<i>basic</i>): $\gamma(\mathbf{v}) = [\cos(2\pi\mathbf{v}), \sin(2\pi\mathbf{v})]^T$ y mapeo de características de Fourier gaussiano: $\gamma(\mathbf{v}) = [\cos(2\pi\mathbf{B}\mathbf{v}), \sin(2\pi\mathbf{B}\mathbf{v})]^T$, con $\sigma = 1$, $\sigma = 10$ y $\sigma = 100$. Además se muestra la imagen original o datos verdaderos (Ground truth, GT).	34
3.7. Evolución de la relación señal-ruido (PSNR) a lo largo de las 2000 épocas de entrenamiento de la red para los conjuntos de entrenamiento (<i>train</i>) y prueba (<i>test</i>) que obtuvieron mejor rendimiento (FF = Fourier features).	34
3.8. Resultado de la regresión de una porción de 512×512 píxeles de una las imágenes de nuestro conjunto de datos usando SIREN y características de Fourier (FF). Además se muestra la imagen original o datos verdaderos (Ground truth, GT).	35

4.1. 21 imágenes que escanean la línea fotosférica de absorción del hierro neutro centrada en 6302 Å (Fe I λ 6302). Cada imagen se ha etiquetado como λ_t con t de 1 a 21.	38
4.2. Representación tridimensional del cubo de imágenes que escanean la línea fotosférica de absorción de hierro neutro centrada en 6302 Å (Fe I λ 6302). Las 21 imágenes están equiespaciadas en longitud de onda y etiquetadas como λ_t con t de 1 a 21.	38
4.3. Perfil de la línea de absorción de Fe I λ 6302, que muestra la variación de la intensidad media del píxel a través de las 21 imágenes equiespaciadas. Las intensidades originales de los píxeles se muestran con puntos rojos (<i>original intensities</i>) y en azul una regresión suavizada del cambio de intensidad media a través de la linea de absorción (<i>smoothed mean intensity</i>)	39
4.4. Configuración de la red neuronal para el experimento con SIREN. La entrada está formada por las coordenadas espaciales y de longitud de onda de cada imagen (\mathbf{x}_l, λ_l) normalizadas a la rejilla unidad tridimensional, consta de 3 capas ocultas de 256 neuronas y la capa de salida $\mathcal{I}(\mathbf{x}_l)$, que representa la intensidad del píxel para las coordenadas de entrada.	40
4.5. Reconstrucción de una porción de 512×512 píxeles de cuatro imágenes del conjunto de datos utilizando la arquitectura SIREN. Las imágenes superiores representan las reconstrucciones utilizando un valor de $\omega_0 = 50$ y las inferiores las originales (Ground truth, GT).	41
4.6. Evolución de la PSNR para los datos de entrenamiento (<i>train</i>) y prueba (<i>test</i>) con diferentes valores de ω a lo largo de las 2000 épocas (<i>epochs</i>).	42
4.7. Configuración de la red neuronal. La entrada está formada por las coordenadas espaciales y de longitud de onda de cada imagen (\mathbf{x}_i, λ_i) normalizadas a la rejilla unidad tridimensional. Posteriormente se mapean estas coordenadas para obtener el vector de características de Fourier (Ec. 3.10) y la primera capa con k neuronas. A continuación se distribuyen el resto de capas ocultas de 256 neuronas cada una y, por último, la capa de salida que contiene el valor de la intensidad del píxel aprendido por la red.	42
4.8. Evolución de la PSNR para los datos de entrenamiento (<i>train</i>) y prueba (<i>test</i>) con diferentes valores de σ y k a lo largo de las 2000 épocas (<i>epochs</i>).	46
4.9. Reconstrucción de una porción de 512×512 píxeles de cuatro imágenes del conjunto de datos utilizando la arquitectura de <i>Fourier features</i> . Las imágenes superiores representan las reconstrucciones utilizando los valores de $\sigma = [10, 10, 0.8]$, $k = 512$ y las inferiores las originales (Ground truth, GT).	47
4.10. Evolución de la PSNR para los datos de entrenamiento (<i>train</i>) y prueba (<i>test</i>) de los modelos de SIREN con $\omega_0 = 30$ y $\omega_0 = 50$ y de <i>Fourier features</i> con $\sigma = [10, 10, 0.8]$ y los valores de $k = 512$ y $k = 128$ a lo largo de las 2000 épocas (<i>epochs</i>).	48

4.11. Comparativa de la evolución de la intensidad media de píxeles entre las imágenes generadas por los modelos de SIREN y de <i>Fourier features</i> , junto con los puntos correspondientes a las medias de intensidad de cada una de las imágenes reales (GT)	48
4.12. Imágenes generadas correspondientes a las longitudes de onda intermedias entre λ_9 y λ_{11} . La selección de este rango se basa en la intersección de las curvas de intensidad de los modelos en la Fig. 4.11, lo que proporciona un caso de prueba para examinar la capacidad de interpolación espectral y la fidelidad de la reconstrucción.	49
A.1. Vista de tensorboard para el resumen de ejecución del modelo de Fourier features utilizando la GPU A100 en Google Colab.	66
A.2. Vista de tensorboard del uso de la memoria asignada y reservada del modelo de Fourier features utilizando la GPU A100 en Google Colab.	67

Capítulo 1

Introducción

Las líneas de absorción que observamos en el Sol son potentes herramientas de diagnóstico de la estratificación del medio donde se originan. Estas líneas, también conocidas como el espectro de Fraunhofer, son resultado directo de la interacción entre la luz emitida por el Sol y los diferentes elementos presentes en su atmósfera. Cuando la luz atraviesa estas capas de gas caliente, ciertas longitudes de onda específicas son absorbidas por los electrones de sus átomos, impidiendo que lleguen hasta nosotros. Cada elemento químico posee un conjunto único de líneas de absorción a longitudes de onda específicas que corresponden a los niveles energéticos particulares de sus electrones. La detección e identificación de estas líneas (Fig. 1.1) nos permite, por lo tanto, reconocer los elementos presentes y entender las condiciones físicas y los procesos que tienen lugar en el Sol.

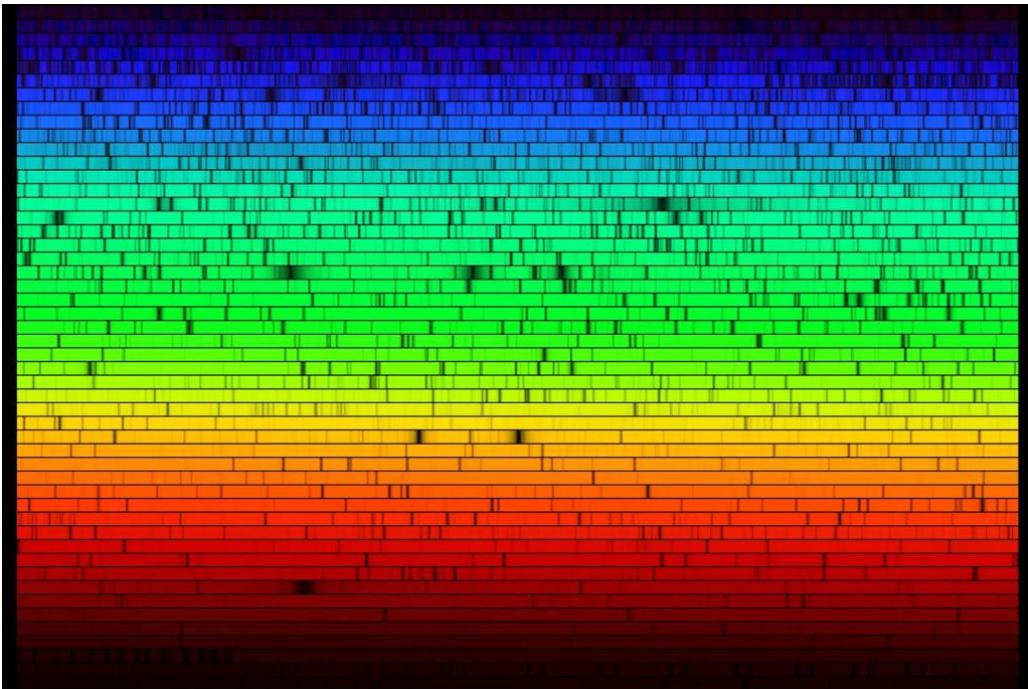


Figura 1.1: Líneas espectrales de absorción en el espectro solar.

Las líneas son más oscuras que su entorno ya que la temperatura al nivel de la atmósfera donde se forman es inferior a la temperatura de 5800 K en la base de la fotosfera, donde se origina la mayor parte de la emisión continua. De hecho, también puede entenderse la formación de una línea de absorción en términos de la disminución de la temperatura de la función fuente local (cantidad de luz que se emite desde un punto específico y en una dirección concreta) hacia el centro de la línea, tal y como muestra la Fig. 1.2. A medida que avanzamos hacia la superficie, la línea se hace cada vez más fuerte, esto es, más profunda. Su evolución del fondo a la superficie sería similar a la representada en la Fig. 1.3.

La opacidad de este medio, κ_ν , puede depender de la longitud de onda, y da cuenta de su capacidad de *parar* los fotones incidentes que provienen de zonas más profundas del Sol. Para

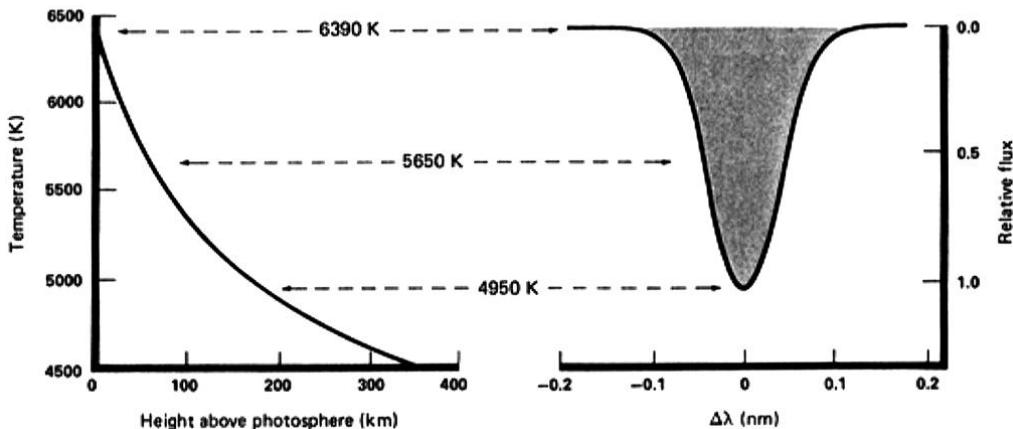


Figura 1.2: Variación de la temperatura en la fotosfera y formación de una línea de absorción.

el caso de una absorción uniforme de la intensidad incidente, I_0 , a lo largo de una distancia determinada dentro del medio:

$$I = I_0 e^{\tau_v}, \quad (1.1)$$

con τ_v la profundidad óptica, que depende de κ_ν . Cuanto menor sea τ_v , más profundo veremos a través del medio.

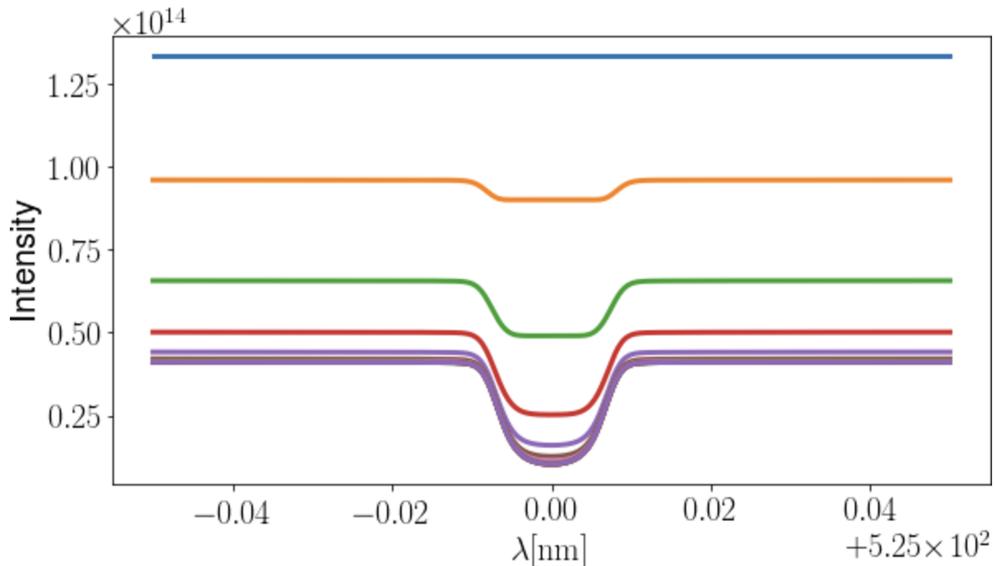


Figura 1.3: De arriba a abajo, variación de la cantidad de absorción a medida que ascendemos en la fotosfera solar.

Bajo la fotosfera solar, el gas es lo suficientemente denso y las interacciones entre fotones, electrones e iones lo suficientemente comunes para que la radiación no pueda escapar directamente al espacio. En la fotosfera, sin embargo, la probabilidad de que un fotón escape sin más interacción con la materia depende de su energía. Si esa energía corresponde a alguna transición electrónica de uno de los átomos o iones presentes en el gas, entonces el fotón puede ser absorbido antes de que pueda viajar más lejos: cuantos más elementos del tipo adecuado para la absorción existan, menor será la probabilidad de escape. Por el contrario, si la energía del fotón no coincide con ninguna de esas transiciones, entonces el fotón no podrá interactuar más con el gas y puede escapar del Sol. Por lo tanto, como ilustra la Fig. 1.4, cuando miramos al Sol,

en realidad estamos mirando hacia adentro en su fotosfera, a una profundidad que depende de la longitud de onda de la luz considerada. Los fotones con longitudes de onda muy alejadas de cualquier absorción característica tienden a provenir de las profundidades de la fotosfera, mientras que aquellos que forman los centros de las líneas de absorción —con mayor probabilidad de interacción con la materia a medida que viajan a través del gas solar— escapan principalmente de los niveles más externos y fríos.

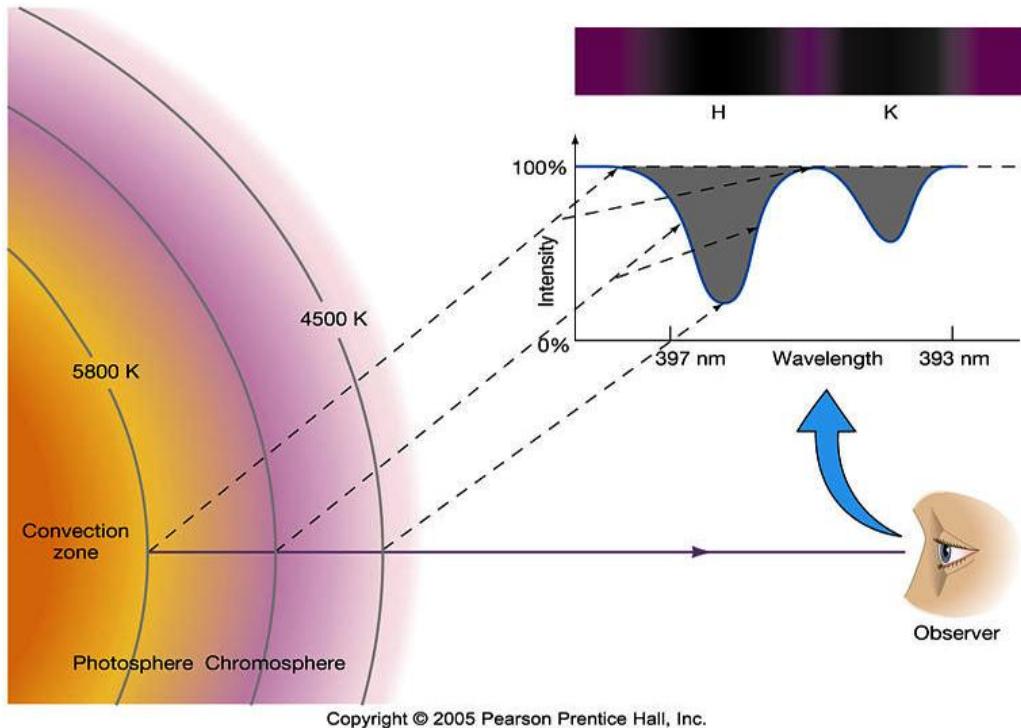


Figura 1.4: Esquema de la formación de las líneas H y K de absorción del Ca II. Los fotones con energías cercanas a una transición electrónica tienen mayor probabilidad de ser absorbidos antes de escapar, por lo que los que llegan a la Tierra provienen de niveles más altos y fríos de la fotosfera solar.

1.1. Espectroscopía tridimensional del Sol

Consideremos ahora una imagen (artificial) de una región determinada del Sol (Fig. 1.5), y supongamos que para cada píxel de la imagen tenemos también su espectro, es decir, disponemos de un conjunto de datos en 3D, $I(x, y, \lambda)$, con λ la longitud de onda. La pregunta es: ¿qué nos dice la diferencia entre el espectro de un píxel más brillante (zona más caliente) y el de uno más oscuro (más frío) en las imágenes? Los distintos niveles de intensidad a diferentes longitudes de onda nos hablan de la estratificación de la temperatura en la fotosfera solar. Además, los gránulos que vemos en la Fig. 1.5 son más calientes cuanto más profundos, y más fríos hacia la superficie. Por lo tanto, usando métodos de inversión podríamos recuperar la variación de diferentes parámetros físicos con la profundidad como la composición química, la temperatura, la presión o la densidad.

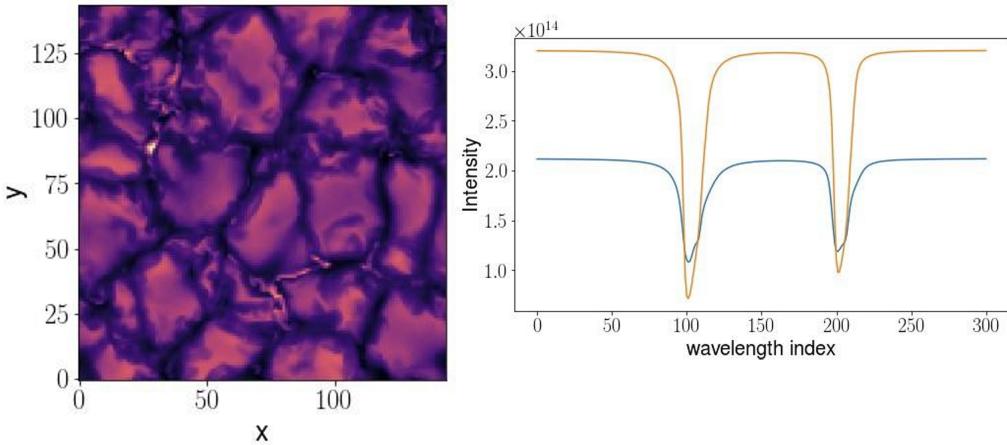


Figura 1.5: Imagen sintética de una región del Sol donde se observa la granulación solar (izquierda). Para cada uno de sus píxeles también se dispone de un espectro (derecha). Como ejemplo se muestra el espectro, que contiene dos líneas de absorción, de un píxel brillante (amarillo) y el de uno oscuro (azul).

1.2. Motivación y objetivos

La obtención de imágenes precisas a diferentes longitudes de onda es una necesidad fundamental en múltiples campos de la ciencia, desde la astrofísica hasta la biomedicina. Sin embargo, las técnicas tradicionales generalmente nos proporcionan datos espectrales en longitudes de onda discretas. Con este PFG buscamos llenar las lagunas entre estas longitudes de onda discretas para obtener una información más completa y detallada. En particular, en este proyecto nos centraremos en obtener una interpolación espectral entre imágenes obtenidas a diferentes longitudes de onda de un área específica de la superficie del Sol. Estas imágenes se han obtenido con la ayuda de un interferómetro de Fabry-Perot (Fig.1.6).

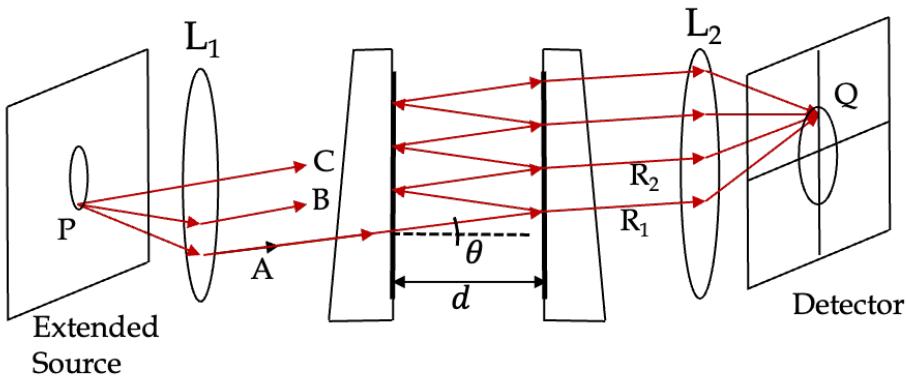


Figura 1.6: Configuración de un interferómetro de Fabry-Perot. Los rayos que provienen de un punto de la fuente P emergen paralelos tras la lente convergente L_1 . Estos rayos A , B , C entran con el mismo ángulo de inclinación θ en el espacio entre los dos reflectores parciales separados por una distancia d . A su salida, los rayos (R_1 , R_2 , ...) se dirigen al plano focal de la lente L_2 y convergen en el mismo punto Q de la pantalla o detector.

Un interferómetro de Fabry-Perot (etalón) [1] es un dispositivo óptico que se utiliza en espectrografía solar para analizar la luz emitida por el Sol y obtener información detallada sobre su composición, temperatura, densidad, velocidad de rotación y otras magnitudes físicas a partir de espectros. Consta de dos espejos paralelos, que forman una cavidad donde la luz se refleja

repetidamente, creando múltiples haces que interfieren entre sí. Esta interferencia produce un patrón que puede analizarse para determinar su longitud de onda.

Una característica clave de los etalones es su alta resoluciónpectral, lo que significa que pueden distinguir entre longitudes de onda de luz muy cercanas entre sí. Los espectrógrafos de filtro de los telescopios solares, como es el caso del espectrógrafo TESOS *Triple Etalon Solar Spectrograph* [2, 3] en el observatorio del Teide, los utilizan como filtros estrechos para proporcionar imágenes pseudomonocromáticas de alta resolución que se obtienen tomando medidas en puntos específicos y predefinidos a lo largo de una líneaespectral.

Dependiendo de la configuración en que se usen, pueden producir desviaciones de la longitud de onda del filtro en cada píxel. Generalmente, estos efectos se intentan corregir mediante calibraciones, pero una interesante posibilidad a explorar es aprovechar estos desplazamientos para producir artificialmente imágenes con una mayor resoluciónspectral, como si se hubiesen usado muchos más filtros estrechos a lo largo de la misma línea de absorción. Para ello, es necesario en primer lugar estudiar la posibilidad de usar el *machine learning* para aproximar las imágenes observadas. Haremos esto usando *neural fields*, también conocidos como *implicit neural representations* [4].

Un *neural field* aplicado a una imagen pixelada consiste en una red neuronal que tiene como entradas las coordenadas (x, y) del píxel y como salida su valor RGB (o cualquier otra codificación). Cuando se pregunta a un *neural field* entrenado que dé la salida para todos los píxeles, se reconstruye la imagen. Un *neural field* es, por tanto, una función continua y derivable que aproxima una función definida en un espacio, en nuestro caso $I(x, y)$.

En los últimos años, han surgido técnicas modernas como las *SIREN* [5] (*SINusoidal REpresentation Networks*) y las *Fourier Features* [6] que han demostrado obtener muy buenos resultados en la regresión de imágenes bidimensionales a partir de sus coordenadas espaciales. Las publicaciones en las que se presentan describen un rendimiento superior para esta tarea concreta en comparación con otras técnicas, particularmente en términos de la métrica de la relación señal-ruido (PSNR) usada para determinar la calidad de la reproducción de las imágenes [7]. Partiremos de estos experimentos que ya han probado su eficacia y los reproduciremos con nuestros datos para, posteriormente, intentar abordar el problema de la interpolaciónspectral.

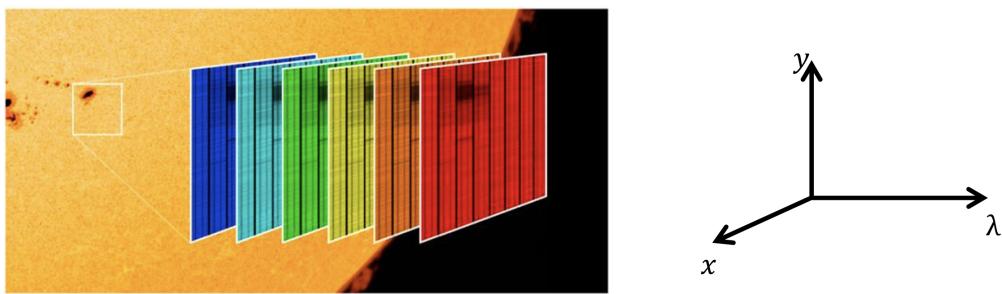


Figura 1.7: Ejemplo ilustrativo de un cubo de seis imágenes de una región de la fotosfera solar en seis longitudes de onda distintas.

Para este PFG disponemos de un cubo de datos espectrales reales $I(x, y, \lambda)$ de la líneafotosférica de absorción del hierro neutro (esto es, no ionizado) centrada en 6302 Å (Fe I λ 6302). Los datos proceden del *Swedish Solar Telescope*, en La Palma y fueron recogidos mediante el

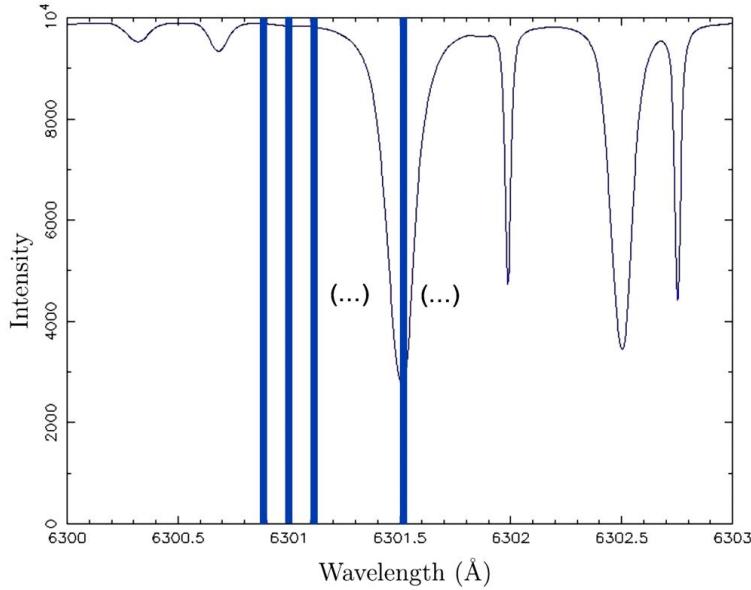


Figura 1.8: Cada línea vertical indica alguna de las 21 bandas espectrales pseudomonocromáticas en las que se escanea la línea de absorción de Fe I $\lambda 6302$.

instrumento CRISP (*CRisp Imaging SpectroPolarimeter*) [8]. En particular, son 21 imágenes pseudomonocromáticas de 966×964 píxeles cada una de un área grande de la superficie del Sol. La resolución del instrumento es de $0.057 \text{ arcsec/pixel}$. Un *arcsec* equivale, en la superficie del Sol, a unos 725 km . Así que cada imagen es de aproximadamente $55.1 \times 54.9 \text{ arcsec}$, que son unos $39.9 \times 39.8 \text{ Mm}$. Estas imágenes *escanean* la línea de absorción de Fe I $\lambda 6302$ en 21 intervalos de longitud de onda estrechos. Grossó modo, estamos ante un esquema similar al que ilustra la Fig. 1.7, que presenta seis imágenes desde el azul al rojo, pero en nuestro caso centrado en una única línea de absorción de la fotosfera del Sol. La Fig. 1.8 muestra un esquema del escaneo de la línea de absorción del hierro neutro a 6302 Å , y la Fig. 1.9 tres de las 21 imágenes con las que trabajaremos.

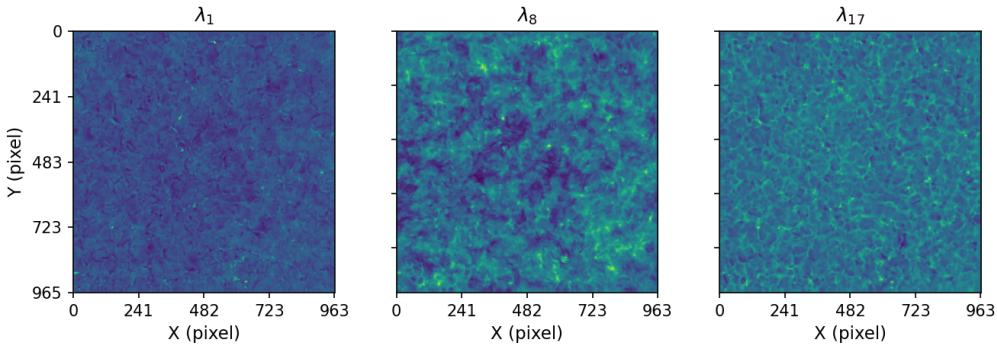


Figura 1.9: Tres de las 21 imágenes del escaneo de la línea de absorción de Fe I $\lambda 6302$ en la misma región del Sol. Nótense las diferentes intensidades en distintas partes de la línea.

El primer intento será verificar la arquitectura que mejor aproxime las imágenes pseudomonocromáticas. Si esto se consigue, daremos el paso a extender esta aproximación al espacio completo incluyendo la longitud de onda, es decir, $I(x, y, \lambda)$. Esto permitirá verificar que es posible compactar el cubo de 21 imágenes en los pesos de una red neuronal. El entrenamiento es relativamente sencillo: solo requiere redes de tipo *fully connected* y optimizadores de tipo *stochastic gradient*, ampliamente conocidos en el campo del *deep learning*. Tal experimento

tiene un recorrido limitado, salvo el hecho de que podemos interpolar entre imágenes con total libertad. El objetivo final es verificar si podemos usar esta aproximación para llegar a la superresolución usando un *neural field*.

Capítulo 2

Campos neuronales

2.1. Campos

El concepto de campo en física fue desarrollado principalmente por Michael Faraday en el siglo XIX durante sus trabajos experimentales en electromagnetismo. Faraday introdujo la idea de líneas de fuerza para representar los campos eléctrico y magnético [9]. Esta representación visual proporcionó una forma intuitiva de entender cómo las fuerzas podrían transmitirse a través del espacio vacío, lo que contradecía las teorías dominantes de la época que sostenían que las fuerzas solo podían transmitirse mediante el contacto directo.

La visión de Faraday de los campos fue formalizada por James Clerk Maxwell en sus famosas ecuaciones, presentadas en 1864 [10], que describen cómo funcionan los campos eléctricos y magnéticos y sobre las que se fundamenta la teoría electromagnética clásica.

El concepto de campo en física se ha extendido más allá del electromagnetismo para abarcar una amplia variedad de fenómenos físicos. Ahora incluye campos gravitatorios (descritos por la teoría de la relatividad general de Einstein [11]) y campos cuánticos que subyacen en la física de partículas. Dentro de la teoría cuántica de campos, impulsada por Dirac [12], se describen las interacciones fuertes y débiles, las cuales son fundamentales en la física de partículas y están representadas por sus respectivos campos cuánticos. En todos estos contextos, un campo se entiende generalmente como una entidad física que tiene un valor en cada punto del espacio y del tiempo.

La idea de campo se aborda en innumerables libros de física pero, como síntesis conceptual, podemos describirlo como una función que asigna a cada punto del espacio (y quizás del tiempo) un valor de alguna magnitud física. Es decir, describe la manera en que dicha magnitud cambia o se distribuye en el espacio y en el tiempo. La magnitud física puede ser cualquier propiedad medible, como la temperatura, la velocidad, la densidad, la presión, la carga eléctrica, y por qué no, la intensidad de un píxel. Dependiendo de la cantidad física que representan, los campos pueden ser escalares, vectoriales o tensoriales.

Los campos, por lo tanto, son una herramienta fundamental para describir cómo las magnitudes físicas varían en el espacio y el tiempo, y cómo afectan a las partículas y a los objetos que se encuentran en su entorno.

2.2. Redes neuronales

Las redes neuronales constituyen un pilar fundamental en el auge del campo de la inteligencia artificial y el aprendizaje automático. Inspiradas en el funcionamiento del cerebro humano, las

redes neuronales son paradigmas de computación diseñados para imitar la forma en que los humanos aprenden y procesan la información. Desde su concepción en la década de 1940, estas redes han evolucionado considerablemente, superando a los algoritmos convencionales en una serie de tareas complejas y variadas, que abarcan desde el procesamiento del lenguaje natural [13] hasta el reconocimiento de imágenes [14] y la generación de texto [15].

Una red neuronal está compuesta por una serie de elementos de cálculo, conocidos como neuronas. Estas se organizan en diferentes capas y se fundamentan en el modelo matemático de neurona propuesto por Warren McCulloch y Walter Pitts en 1943 [16].

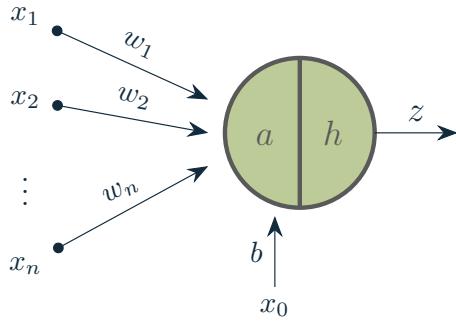


Figura 2.1: Modelo neuronal de McCulloch-Pitts en el que se incluye el conjunto de datos de entrada x_1, x_2, \dots, x_n y sus pesos asociados w_1, w_2, \dots, w_n . La letra a corresponde a la suma ponderada de las entradas más el sesgo b . La letra h representa la función de transferencia o activación que se aplica a la suma ponderada para obtener la salida z .

Cada neurona recibe un conjunto de entradas $\{x_1, x_2, \dots, x_N\}$, realiza cálculos y pasa la salida a las neuronas de la capa siguiente. Las entradas a una neurona pueden ser los datos brutos de un problema o las salidas de las neuronas en las capas anteriores. La salida z se calcula aplicando una función de activación h a la suma ponderada de sus entradas. Los pesos $\{w_1, w_2, \dots, w_N\}$ que se aplican a las entradas determinan la intensidad de la conexión y son los parámetros que la red aprende durante el entrenamiento:

$$z = h(a) = h \left(\sum_{i=1}^N x_i w_i \right) . \quad (2.1)$$

Aunque originalmente no se incluía, en la mayoría de los modelos modernos de redes neuronales se añade un término de sesgo a cada neurona. El sesgo, representado como b en la Fig. 2.1, es un parámetro que se suma después de realizar la suma ponderada de las entradas y proporciona una forma adicional de adaptar la salida de la neurona, lo que puede mejorar la capacidad del modelo para representar datos y realizar tareas de clasificación o predicción más precisas:

$$z = h(a) = h \left(\sum_{i=1}^N x_i w_i + x_0 w_0 \right) . \quad (2.2)$$

Si consideramos la entrada $x_0 = 1$ y el sesgo $b = w_0$ en la Fig. 2.1, podemos escribir la ecuación (2.2) de forma más compacta:

$$z = h(a) = h \left(\sum_{i=1}^N x_i w_i + b \right) = h \left(\sum_{i=0}^N x_i w_i \right) . \quad (2.3)$$

Por último, la función de activación es una función matemática que transforma la suma ponderada de las entradas de una neurona. Introduce no linealidad en el modelo, lo que permite que la red neuronal aprenda y represente relaciones más complejas entre las entradas y las salidas. Actualmente las funciones de activación más frecuentes son las siguientes:

- **Función escalón:** Es la función empleada en el modelo básico de McCulloch-Pitts. Produce una salida binaria (como 0 o 1) en función de un cierto umbral (Fig. 2.2a):

$$h(a) = \begin{cases} 0 & \text{si } a < 0 \\ 1 & \text{si } a \geq 0 \end{cases} \quad (2.4)$$

- **Función sigmoide:** Produce una salida entre 0 y 1. Tiene forma de S y su salida puede interpretarse como una probabilidad (Fig. 2.2b):

$$h(a) = \frac{1}{1 + e^{-a}} \quad (2.5)$$

- **Función tangente hiperbólica (tanh):** Similar a la función sigmoide, pero produce una salida entre -1 y 1. Esto permite que los datos de salida estén centrados en torno al 0, lo que puede mejorar la capacidad de representación y facilita la normalización de los datos (Fig. 2.2c):

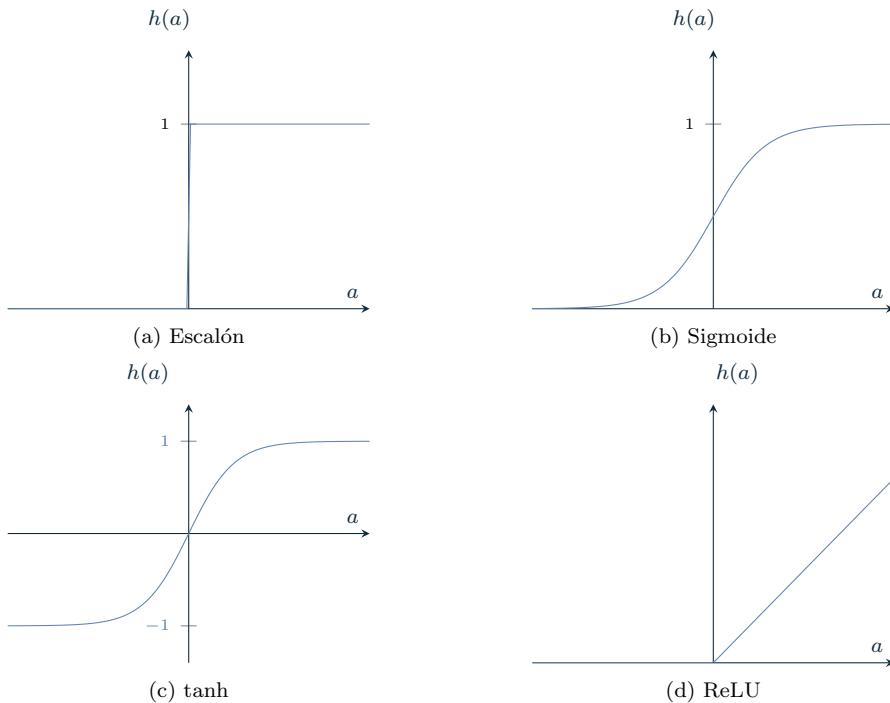
$$h(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}} \quad (2.6)$$

- **Función ReLU (Rectified Linear Unit):** Es la función de activación más utilizada en las redes neuronales convolucionales y las redes de retropropagación de errores (Fig. 2.2d):

$$h(a) = \begin{cases} 0 & \text{si } a < 0 \\ a & \text{si } a \geq 0 \end{cases} \quad (2.7)$$

En su forma más simple, una red neuronal consta de tres capas: una capa de entrada, una capa oculta y una capa de salida (Fig 2.3). La **capa de entrada**, que no realiza ningún procesamiento en sí misma, transfiere la información a la siguiente capa, conocida como capa oculta. El número de nodos (neuronas) en la capa de entrada generalmente corresponde al número de características o atributos de los datos de entrada. La adecuación de estos datos para la red es uno de los aspectos más críticos del aprendizaje profundo, dado que la calidad y la presentación de los datos de entrada pueden influir significativamente en el rendimiento del modelo.

La **capa oculta** es el núcleo computacional de la red neuronal, donde se realiza la mayor parte del procesamiento. Compuesta por un número definido de neuronas, la capa oculta procesa los datos recibidos de la capa de entrada. A través de los cálculos realizados en estas capas, la red logra aprender de los datos de entrada y mejora la precisión de sus predicciones. Las redes más complejas y profundas pueden contar con múltiples capas ocultas.

Figura 2.2: Funciones de activación h más comunes.

Finalmente, la **capa de salida** es la encargada de proporcionar el resultado final generado por la red neuronal. Dependiendo del tipo de problema que la red esté diseñada para resolver, este resultado puede ser una clasificación (para problemas de clasificación) o un valor específico (para problemas de regresión). El número de nodos (neuronas) en la capa de salida suele corresponder al número de clases o resultados posibles que la red puede producir.

2.2.1. Entrenamiento de una red neuronal

El entrenamiento de una red neuronal es un proceso iterativo que busca optimizar sus pesos y sus sesgos para que pueda realizar predicciones precisas. Este proceso se realiza usando un conjunto de datos de entrenamiento etiquetados, que proporcionan ejemplos de las entradas y las salidas esperadas. A continuación, se describen los procesos de obtención de resultados (paso hacia adelante) y el ajuste de los pesos y los sesgos (retropropagación) que tienen lugar en cada iteración.

Propagación (forward pass)

En cada iteración del entrenamiento, la red neuronal realiza una serie de cálculos en un proceso conocido como *forward pass* o paso hacia adelante. Durante este paso, las entradas se multiplican por los pesos y se suman los sesgos en cada neurona de las capas ocultas y de salida. Estas sumas ponderadas se pasan luego a través de las funciones de activación correspondientes. Así, las expresiones de salida de las neuronas de la capa oculta de la red neuronal simple que ilustra la Fig. 2.3 en un paso hacia adelante serían:

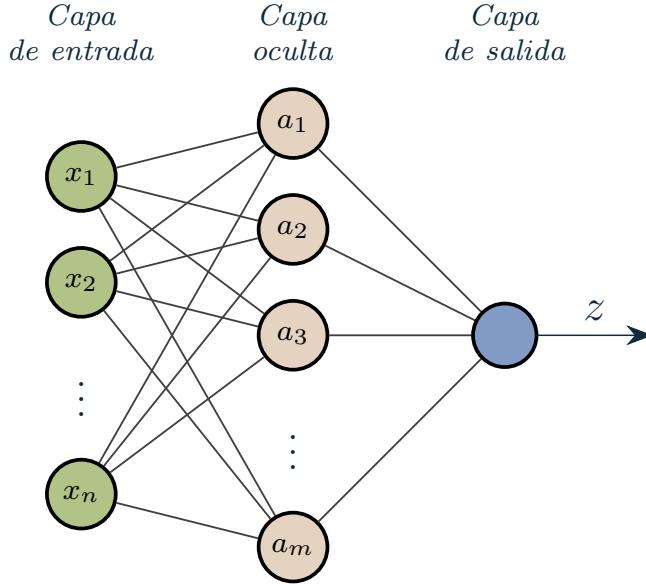


Figura 2.3: Configuración simple de una red neuronal que cuenta con una capa de entrada, una capa oculta y una capa de salida.

$$\begin{aligned}
 z_1 &= h(a_1) = h \left(\sum_{i=1}^n x_i w_{1,i} + b_1 \right) \\
 z_2 &= h(a_2) = h \left(\sum_{i=1}^n x_i w_{2,i} + b_2 \right) \\
 &\vdots \\
 z_m &= h(a_m) = h \left(\sum_{i=1}^n x_i w_{m,i} + b_m \right) ,
 \end{aligned} \tag{2.8}$$

que en forma matricial queda:

$$\begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_m \end{bmatrix} = h \left(\begin{bmatrix} w_{1,1} & w_{1,2} & \dots & w_{1,n} \\ w_{2,1} & w_{2,2} & \dots & w_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{m,1} & w_{m,2} & \dots & w_{m,n} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} \right) , \tag{2.9}$$

y que se puede escribir como:

$$\mathbf{z} = h(\mathbf{W}\mathbf{x} + \mathbf{b}) , \tag{2.10}$$

Este proceso se propaga desde la capa de entrada hasta la capa de salida, produciendo una salida que es la estimación de la red neuronal basada en los datos de entrada. Dicha salida puede expresarse matemáticamente como:

$$z_s = h_s \left(\sum_{i=1}^m z_i w_{s,i} + b_s \right) , \tag{2.11}$$

Es decir:

$$z_s = h_s \left(\begin{bmatrix} w_{s,1} \\ w_{s,2} \\ \vdots \\ w_{s,m} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_m \end{bmatrix} + b_s \right) = h_s(\mathbf{w}_s \mathbf{z} + b_s). \quad (2.12)$$

En el contexto del problema de regresión, donde el objetivo es predecir la intensidad del píxel a partir de sus coordenadas espaciales (x, y) , la diferencia entre el valor de intensidad predicho por la red neuronal y el valor real se conoce como el error para esa muestra específica. Para evaluar la efectividad del modelo en todo el conjunto de entrenamiento se utiliza una función de pérdida. En este caso, una de las más comunes es el error cuadrático medio (*Mean Squared Error*, MSE). Esta métrica representa una penalización del error acumulado en todas las muestras del conjunto de entrenamiento y se busca minimizarlo durante el proceso de entrenamiento:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^N (z_{\text{true},i} - z_{s,i})^2. \quad (2.13)$$

En esta ecuación:

N representa el número total de muestras,

$z_{\text{true},i}$ es el valor verdadero de la etiqueta correspondiente a la i -ésima entrada de datos,

$z_{s,i}$ es la predicción realizada por la red neuronal para esa entrada.

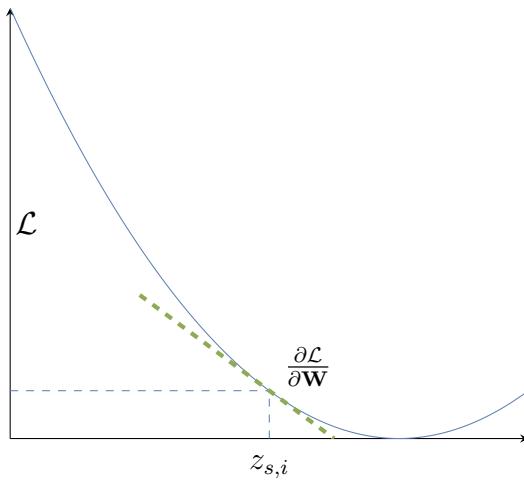


Figura 2.4: Evolución de la función de pérdida $\mathcal{L} = \frac{1}{N} \sum_{i=1}^N (z_{\text{true},i} - z_{s,i})^2$.

Cuando $z_{s,i}$ es igual al valor verdadero $z_{\text{true},i}$, la pérdida es 0, lo que indica que la predicción es correcta. Por el contrario, a medida que la predicción se aleja del valor verdadero, la pérdida aumenta. La pendiente de la función de pérdida en el punto $z_{s,i}$ se utiliza en el algoritmo de descenso de gradiente para actualizar los parámetros del modelo (Fig. 2.4).

Debido al cuadrado en su fórmula, el MSE penaliza con más fuerza las predicciones que se alejan del valor verdadero. No obstante, también puede ser sensible a valores de entrada que difieren significativamente del patrón general de los datos, conocidos como valores atípicos (*outliers*). Por ello, en ocasiones se utiliza el error absoluto medio (*Mean Absolute Error*, MAE), que puede resultar más apropiado si se espera que el conjunto de datos contenga este tipo de valores:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^N |z_{\text{true},i} - z_{s,i}| . \quad (2.14)$$

Aunque el MSE también se utiliza en problemas de clasificación, la función de pérdida más comúnmente usada es la entropía cruzada (*Cross-Entropy Loss*). Esta métrica evalúa el rendimiento de un modelo de clasificación cuyas salidas son probabilidades entre 0 y 1. Para la clasificación binaria, se usa la *Binary Cross-Entropy*, y para la clasificación multiclase, la *Softmax Cross-Entropy*.

Retropropagación (*backpropagation*)

La discrepancia obtenida tras la aplicación de la función de pérdida se emplea para ajustar los pesos y sesgos de la red en una dirección que minimice el error. Este ajuste se lleva a cabo mediante un algoritmo conocido como retropropagación o *backpropagation*, que se apoya en la regla de la cadena del cálculo diferencial para propagar el error desde la salida de la red hacia sus capas anteriores, asegurando así que cada nodo contribuya de manera proporcional a la minimización del error global.

De manera general, se persigue actualizar el valor de los parámetros w y b para minimizar la función de pérdida (\mathcal{L}). Por lo tanto, el proceso de optimización debe permitir alcanzar el mínimo de \mathcal{L} moviéndonos en la dirección del gradiente. Podemos visualizar el proceso de la siguiente manera si reducimos la dimensionalidad del problema: el gradiente nos proporciona la dirección de mayor cambio. De esta forma, actualizaremos nuestros parámetros siguiendo el negativo del gradiente (minimización de la función de pérdida).

Por lo tanto, para actualizar nuestros parámetros w y b seguiremos las siguientes ecuaciones:

$$w_{\text{actualizado}} = w - \alpha \frac{\partial \mathcal{L}}{\partial \mathbf{W}}, \quad (2.15)$$

$$b_{\text{actualizado}} = b - \alpha \frac{\partial \mathcal{L}}{\partial b}, \quad (2.16)$$

donde α es el valor del paso (*step size* o *learning rate*). El valor de este parámetro es crítico, ya que si se escoge muy pequeño se tarda demasiado en alcanzar el mínimo y, por el contrario, si se escoge muy grande puede dar lugar a saltos que nos lleven a la pendiente contraria.

Las derivadas parciales $\partial \mathcal{L}/\partial \mathbf{W}$ y $\partial \mathcal{L}/\partial b$ representan cómo un pequeño cambio en los pesos o en los sesgos puede afectar a la función de pérdida, y se obtienen mediante la aplicación de la regla de la cadena durante el algoritmo de *backpropagation*. La selección adecuada del valor de α requiere cierto ajuste y experimentación, y existen diversas estrategias y algoritmos avanzados de optimización que ajustan este valor de manera adaptativa. Durante el proceso de retropropagación, se calcula el gradiente del error con respecto a cada peso y cada sesgo

de la red. Este gradiente representa la dirección y la magnitud de ajuste de los pesos y los sesgos para minimizar el error. Posteriormente, se efectúa un ajuste a cada peso y cada sesgo proporcional al opuesto de su gradiente, en un proceso conocido como descenso por gradiente o *gradient descent*. Este algoritmo de optimización es un procedimiento que tiene por objetivo minimizar la función de pérdida de manera iterativa hasta que la red converja a una solución. Aunque esta solución no siempre es la óptima global, suele ser suficientemente buena para el objetivo propuesto.

Si consideramos las operaciones que tienen lugar en una neurona y las representamos de forma separada para entender cómo se produce el flujo de los datos, vemos que para una entrada \mathbf{x} y un peso \mathbf{w} se genera una función intermedia que se puede expresar como $\mathbf{t} = \mathbf{w} \cdot \mathbf{x}$. Posteriormente, tras la suma del sesgo b , y la aplicación de la función de activación h , obtenemos la función de salida de la neurona $z = h(\mathbf{t} + b)$.

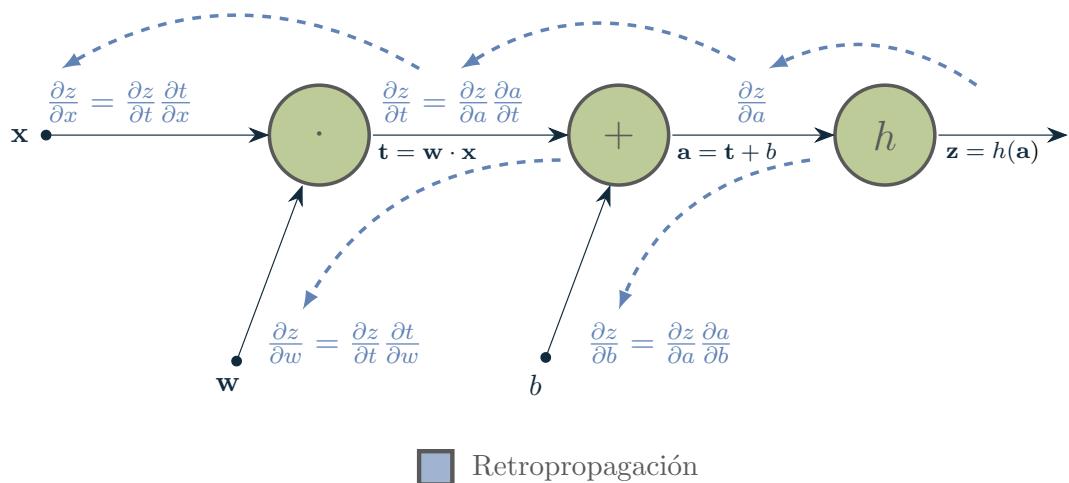


Figura 2.5: Representación de cómo se produce el flujo de datos y la retropropagación en una neurona basada en la regla de la cadena del cálculo diferencial.

A la entrada de cada operación tenemos los gradientes locales:

$$\frac{\partial t}{\partial x}, \frac{\partial t}{\partial w}, \frac{\partial a}{\partial t}, \frac{\partial a}{\partial b} \text{ y } \frac{\partial z}{\partial a}.$$

Aplicando la regla de la cadena obtenemos los gradientes de la función de salida con respecto a cada parámetro. En la Fig. 2.5 se considera que la variable \mathbf{x} de entrada proviene de una neurona anterior y no es el valor del dato de entrada a la red (en ese caso no tendría sentido la retropropagación). Los valores de los gradientes obtenidos nos permitirán actualizar nuestros parámetros mediante las Ec. (2.15) y (2.16).

Este proceso cíclico de ajuste basado en errores es conocido como aprendizaje supervisado. En cada etapa, la red recibe supervisión a través de las etiquetas de salida correctas que se le suministran y utiliza esta supervisión para aprender y optimizarse.

Con la mejora del hardware de computación y el crecimiento de los conjuntos de datos, las redes neuronales se han vuelto más profundas y complejas, lo que ha llevado al desarrollo del aprendizaje profundo. Las redes neuronales profundas pueden tener cientos o incluso miles de capas y son capaces de aprender representaciones muy abstractas y complejas de sus datos.

de entrada. Estos avances han permitido a las redes neuronales resolver tareas que antes se consideraban fuera de su alcance como el procesamiento del lenguaje natural, el reconocimiento de imágenes o la detección de enfermedades o identificación de patrones anómalos en resonancias magnéticas o tomografías computarizadas.

2.3. Campos neurales (*neural fields*)

El teorema de aproximación universal, demostrado por primera vez por George Cybenko en 1989, establece que una red neuronal con una sola capa oculta puede aproximar cualquier función continua en conjuntos cerrados y acotados de \mathbb{R}^n , dadas ciertas condiciones sobre la función de activación de la neurona [17]. Este teorema fue posteriormente extendido por Kurt Hornik en 1991, quien demostró que el resultado es válido para cualquier función de activación no constante, acotada y monótona creciente [18]. Con esta propiedad de las redes neuronales, en el campo de la investigación en aprendizaje profundo y visión por computadora se ha introducido recientemente el concepto de campo neural o *neural field*.

Un campo neural es aquel que está parametrizado total o parcialmente por una red neuronal [4].

Los campos neurales representan una potente herramienta de procesamiento y análisis de datos, sobre todo en problemas de regresión. Además, han demostrado ser excepcionales en la modelización de propiedades de imágenes basándose en sus coordenadas espaciales. Esto significa que son capaces de identificar y extraer información relevante de una imagen a partir de la distribución espacial de sus píxeles. Esta característica innovadora permite explorar una gama completamente nueva de aplicaciones, lo que expande significativamente los límites de lo que es posible en el campo del procesamiento y análisis de imágenes en campos como la detección y el reconocimiento de objetos en imágenes [19], la segmentación semántica de imágenes [20] o la mejora de la resolución [21].

En el capítulo siguiente se comprueba la eficiencia de estos campos neurales aplicados a un caso concreto: la reconstrucción de imágenes bidimensionales.

Capítulo 3

Regresión de imágenes bidimensionales

En este capítulo, exploraremos dos técnicas modernas de aprendizaje profundo para la reconstrucción de imágenes bidimensionales: las Redes Implícitas Sinusoidales (SIREN) [5] y las características de Fourier (*Fourier features*) [6]. Ambas han demostrado ser muy efectivas en diversas aplicaciones, incluyendo la generación de imágenes [5, 6] y la inferencia de escenas tridimensionales a partir de imágenes bidimensionales [5, 6].

En el contexto de nuestro proyecto, aplicaremos estas técnicas a varios fragmentos de las 21 imágenes pseudomonocromáticas de la superficie solar que describimos en el capítulo 1. Nuestro objetivo no se limita a reproducir la eficacia demostrada de estas técnicas de aprendizaje profundo en la reconstrucción de imágenes. Iremos más allá y demostraremos que, efectuando los ajustes necesarios para adaptarlas a nuestro conjunto de datos, podemos obtener resultados igualmente satisfactorios y avanzar en nuestro camino hacia la superresolución espectral.

3.1. Conceptos previos

Antes de sumergirnos en estas técnicas, es esencial introducir algunos aspectos que son fundamentales en el campo del aprendizaje profundo y que desempeñan un papel clave. Estos aspectos incluyen la inicialización de parámetros, el ajuste de hiperparámetros, los algoritmos de optimización y el preprocesado de los datos. Cada uno de estos elementos contribuye a mejorar el rendimiento y la eficacia de nuestros modelos. A continuación, proporcionamos una visión intuitiva de estos conceptos para sentar las bases necesarias para la discusión sobre SIREN y las características de Fourier (*Fourier features*).

3.1.1. La inicialización de los parámetros

Los pesos y los sesgos, que se ajustan para minimizar la función de pérdida (Ec. 2.13), constituyen los parámetros de la red y su inicialización es un paso crítico en el entrenamiento de las redes neuronales. Los parámetros se configuran inicialmente con valores aleatorios o se determinan mediante alguna regla predefinida. La elección de estos valores iniciales puede tener un impacto significativo en el proceso de entrenamiento y en la eficacia del modelo final [22]. Una inicialización inadecuada puede conllevar un desvanecimiento del gradiente, lo que obstaculiza el aprendizaje.

El desvanecimiento del gradiente es uno de los problemas más habituales durante el proceso de entrenamiento de las redes neuronales profundas, y se produce como consecuencia de la reducción del valor absoluto del gradiente de la función de pérdida a medida que el error se propaga hacia atrás desde las últimas capas de la red. Durante la retropropagación, los gradientes se calculan multiplicando las derivadas parciales del error respecto a cada peso en la red (Fig. 2.5).

Si consideramos una red neuronal con funciones de activación sigmoidales o hiperbólicas, las derivadas de estas funciones estarán en el rango $(0, 0.25)$ y $[0, 1]$, respectivamente. Por lo tanto, durante la retropropagación se van multiplicando entre sí valores cada vez más pequeños y el gradiente puede disminuir exponencialmente [23] a medida que retrocedemos a través de las capas. Cuando el gradiente se aproxima a cero, la actualización de los parámetros de las primeras capas es cada vez más insignificante, lo que da lugar a un aprendizaje lento o casi nulo. Como resultado, el proceso de entrenamiento se vuelve ineficiente y el rendimiento de la predicción significativamente menor.

A continuación, proporcionamos una descripción básica de una serie de técnicas de inicialización de parámetros con el objetivo de ofrecer una visión general de sus diferencias y usos más comunes en el campo del aprendizaje profundo:

- **Inicialización Xavier-Glorot:** esta técnica de inicialización, propuesta en 2010, se aplica antes del inicio del entrenamiento y tiene como objetivo asegurarse de que, en promedio, los valores que pasan por las neuronas durante la propagación (activaciones) y durante la retropropagación (gradientes) mantengan una varianza constante entre las diferentes capas de la red. Al hacerlo, esta técnica contribuye a mitigar el problema del desvanecimiento del gradiente. Para conseguir este objetivo, los pesos W_{ij} en la matriz de pesos \mathbf{W} se inicializan siguiendo una de las dos distribuciones: una normal centrada en cero con varianza $\frac{1}{n}$:

$$W_{ij} \sim \mathcal{N}\left(0, \frac{1}{n}\right), \quad (3.1)$$

o una distribución uniforme en el rango $\left[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}\right]$:

$$W_{ij} \sim \mathcal{U}\left[-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}\right], \quad (3.2)$$

donde n es el número de entradas a la neurona.

- **Inicialización He:** desarrollada en 2015, esta técnica se diseñó para ser más efectiva en combinación con funciones de activación asimétricas como la ReLU (Fig. 2.2d). Aunque su objetivo principal es optimizar la varianza de las activaciones, un efecto secundario beneficioso es que contribuye a reducir el problema del desvanecimiento del gradiente, especialmente en las capas iniciales de redes neuronales profundas. Esto se logra estableciendo la varianza de los pesos en $\frac{2}{n}$, donde n es el número de entradas a la neurona. Los pesos se pueden inicializar usando una distribución normal:

$$W_{ij} \sim \mathcal{N}\left(0, \frac{2}{n}\right), \quad (3.3)$$

o una distribución uniforme:

$$W_{ij} \sim \mathcal{U}\left[-\sqrt{\frac{6}{n}}, \sqrt{\frac{6}{n}}\right]. \quad (3.4)$$

- **Inicialización Normal:** esta estrategia asigna a los pesos valores aleatorios provenientes de una distribución normal estándar, es decir,

$$W_{ij} \sim \mathcal{N}(0, 1). \quad (3.5)$$

Aunque sencillo, este método de inicialización no toma en cuenta el número de entradas o salidas de la neurona. Esto puede llevar a problemas de desvanecimiento del gradiente, especialmente en redes profundas. En particular, la falta de ajuste de la varianza según el número de entradas o salidas de la neurona puede resultar en gradientes que se reducen muy rápidamente durante la retropropagación, impidiendo así el aprendizaje efectivo de la red.

- **Inicialización Uniforme:** Esta estrategia asigna valores aleatorios a los pesos a partir de una distribución uniforme en un intervalo específico, comúnmente $[-1, 1]$:

$$W_{ij} \sim \mathcal{U}(-1, 1) , \quad (3.6)$$

o $[0, 1]$:

$$W_{ij} \sim \mathcal{U}(0, 1). \quad (3.7)$$

Al igual que la inicialización normal, esta técnica puede contribuir al desvanecimiento del gradiente, especialmente en redes profundas, debido a que no ajusta la varianza de los pesos en función del número de entradas o salidas de la neurona.

3.1.2. El ajuste de los hiperparámetros

Los hiperparámetros son variables que definen la estructura del modelo de aprendizaje automático y que influyen en su proceso. Son distintos de los parámetros del modelo, que se aprenden durante el entrenamiento a partir de los datos de entrada. Los hiperparámetros, en cambio, se establecen antes del entrenamiento y, generalmente, no cambian durante este. La selección adecuada de los hiperparámetros puede ser crucial para el rendimiento del modelo, y su ajuste es uno de los desafíos en el aprendizaje automático. Algunos de los hiperparámetros más comunes en las redes neuronales se describen a continuación:

- **Tasa de aprendizaje α (*learning rate*):** este es quizá el hiperparámetro más relevante. Es un factor de escala que determina cuánto se ajustan los pesos en cada actualización durante el entrenamiento (Ec. 2.15 y 2.16). Una tasa de aprendizaje demasiado alta puede hacer que el aprendizaje sea inestable y que el modelo salte sobre el mínimo de la función de pérdida, mientras que una tasa de aprendizaje demasiado baja puede hacer que el aprendizaje sea demasiado lento o que el modelo se quede atascado en mínimos locales.
- **Número de capas ocultas y número de neuronas por capa:** estos hiperparámetros determinan la arquitectura de la red neuronal. En general, las redes con más capas y/o más neuronas son capaces de modelizar funciones más complejas, pero también son más propensas al sobreajuste y requieren más datos para ser entrenadas de manera efectiva.
- **Función de activación:** como vimos en el apartado 2.2, la función de activación determina la salida de una neurona en función de su entrada. Algunas funciones de activación comunes incluyen la función sigmoide (Fig. 2.2b), la tangente hiperbólica (Fig. 2.2c) o la función de activación lineal rectificada (ReLU) (Fig. 2.2d). La elección de la función de activación puede tener un impacto significativo en el rendimiento de la red.
- **Ciclo de entrenamiento (*epoch*):** se define como un recorrido de todos los datos de entrenamiento a través de la red neuronal durante el proceso de aprendizaje. De esta

manera, un ciclo (*epoch*) se completa cuando cada ejemplo de entrenamiento ha sido presentado a la red neuronal una vez. La cantidad de ciclos determina cuántas veces se recorrerá todo el conjunto de datos de entrenamiento. El número óptimo de ciclos (*epochs*) generalmente se obtiene a través de la experimentación y puede variar dependiendo del problema y el conjunto de datos específicos.

- **Tamaño del lote (*batch size*) o mini lote:** El tamaño del lote se refiere a la cantidad de ejemplos de entrenamiento utilizados en una iteración del algoritmo de optimización. Un lote es simplemente un subconjunto del conjunto de datos de entrenamiento. Un tamaño de lote más pequeño puede llevar a una convergencia más rápida, pero también puede hacer que el aprendizaje sea más inestable. Un tamaño de lote más grande puede proporcionar estimaciones del gradiente más precisas, pero también puede hacer que el aprendizaje sea más lento. Un ciclo de entrenamiento (*epoch*) se completa cuando todos los lotes han sido usados una vez para actualizar los pesos.

3.1.3. Los algoritmos de optimización

Los algoritmos de optimización son procedimientos iterativos que buscan minimizar (o maximizar) una función objetivo. Como vimos en el capítulo 2, en el contexto del aprendizaje profundo la función objetivo es típicamente una función de pérdida (ecuación 2.13) que cuantifica la discrepancia entre las predicciones del modelo y los datos de entrenamiento reales. El objetivo de la optimización es encontrar los parámetros que minimizan esta función:

- **Descenso de Gradiente (GD, por sus siglas en inglés):** es el algoritmo de optimización más simple (apartado 2.2.1) y se utiliza ampliamente en el aprendizaje automático y el aprendizaje profundo. En cada iteración, el GD actualiza los parámetros del modelo en la dirección opuesta del gradiente de la función de pérdida con respecto a los parámetros actuales siguiendo las Ec. 2.15 y 2.16.
- **Descenso de Gradiente Estocástico (SGD, por sus siglas en inglés):** Es una variante del GD que estima el gradiente de la función de pérdida utilizando un subconjunto aleatorio de los datos de entrenamiento, conocido como mini lote, en cada iteración. Esto hace que el SGD sea computacionalmente más eficiente que el GD, especialmente para grandes conjuntos de datos.
- **Métodos de optimización adaptativos:** estos algoritmos, como *Adagrad* [24], *RMSprop* [25], *Adam* [26], entre otros, modifican la tasa de aprendizaje durante el entrenamiento. Normalmente, la disminuyen a medida que el entrenamiento avanza e incluso la escalan en función de otros parámetros e hiperparámetros de la red, lo cual puede ayudar a mejorar la convergencia y la estabilidad del entrenamiento.

La elección del algoritmo de optimización puede tener un impacto significativo en el rendimiento de los modelos de aprendizaje profundo y es una parte importante del proceso de ajuste. Sin embargo, no hay una respuesta única a la pregunta de qué algoritmo de optimización es el mejor, ya que la elección óptima puede depender del problema específico, el modelo y los datos [27, 28].

3.1.4. La presentación de los datos de entrada

Los conjuntos de datos desempeñan un papel fundamental en el campo del aprendizaje automático y la inteligencia artificial en las fases de desarrollo y de evaluación de los algoritmos y los modelos [29]. La construcción, el manejo y el análisis de estos conjuntos de datos constituyen una de las partes más relevantes de cualquier proyecto de ciencia de datos, dado que su calidad y representatividad pueden afectar directamente a la efectividad y la precisión de los modelos desarrollados.

El preprocesamiento de los datos, que incluye tareas como la limpieza de los datos, la eliminación de datos irrelevantes o erróneos y la transformación de los datos en una forma adecuada para su análisis, es esencial para garantizar su calidad [30]. Los datos pueden contener ruido, errores, inconsistencias o duplicados que pueden afectar negativamente a la capacidad de un modelo para aprender. El preprocesamiento ayuda a resolver estos problemas y garantiza que los datos sean coherentes y estén bien estructurados, lo que puede facilitar el aprendizaje del modelo y mejorar su rendimiento.

Además, es fundamental dividir el conjunto de datos en conjuntos de entrenamiento, validación y prueba [31, 32]. El conjunto de entrenamiento se utiliza para entrenar el modelo y ajustar sus parámetros. El conjunto de validación se utiliza para afinar el modelo y realizar ajustes, como la selección de hiperparámetros o la detección de sobreajuste (*overfitting*) [32, 33]. Finalmente, el conjunto de prueba se utiliza para evaluar el rendimiento del modelo en datos no vistos previamente y proporcionar una estimación de su capacidad de generalización.

La proporción en que se dividen estos conjuntos de datos puede variar dependiendo de su tamaño y sus características, aunque una división comúnmente utilizada es 70 % para entrenamiento, 15 % para validación y 15 % para prueba [34]. El objetivo de esta división es proporcionar un equilibrio entre maximizar la cantidad de datos disponibles para el aprendizaje y garantizar que se disponga de suficientes datos para validar y evaluar el modelo de manera eficaz.

3.2. La reconstrucción de imágenes bidimensionales

Cada uno de los aspectos tratados en el apartado anterior juegan un papel fundamental en el entrenamiento de las redes neuronales en general y en su aplicación en la reconstrucción de imágenes en particular. A continuación, exploraremos en qué se basan las técnicas SIREN y *Fourier features* centrándonos específicamente en el problema de la regresión de imágenes bidimensionales y los resultados obtenidos al aplicarlas a nuestro conjunto de datos.

3.2.1. Redes implícitas sinusoidales (SIREN)

Las redes implícitas sinusoidales, más conocidas como SIREN [5], son una clase de redes neuronales artificiales que utilizan una función sinusoidal como función de activación. Esta elección de función de activación es única y flexible ya que permite a la red modelizar una gama más amplia de funciones.

Función de activación

Matemáticamente, la activación de una neurona en una SIREN se describe como:

$$h_i = \sin(\boldsymbol{\omega}_i \cdot \mathbf{x} + b_i) , \quad (3.8)$$

donde:

- h_i es la activación de la neurona i ,
- $\boldsymbol{\omega}_i$ es el vector de pesos asociado a la neurona i ,
- \mathbf{x} es el vector de entrada a la neurona, y
- b_i es el sesgo (escalar) de la neurona.

A nivel conceptual, la base de SIREN se encuentra en el análisis de Fourier [35]. Este análisis se fundamenta en la idea de que cualquier función arbitrariamente compleja, como las señales de audio o las imágenes, se puede expresar como una combinación de múltiples funciones sinusoidales. Cada neurona en la red actúa como una onda sinusoidal, oscilando a una frecuencia particular y con una fase específica. Al combinar estas oscilaciones a través de múltiples capas de la red, SIREN puede modelizar funciones de gran complejidad con una gran precisión [5].

Por otro lado, la elección de una función sinusoidal como función de activación tiene varias ventajas adicionales: en primer lugar, es derivable, lo que beneficia enormemente el aprendizaje de la red. Al ser derivable, permite calcular con mayor precisión y suavidad las variaciones en los pesos de la red durante el proceso de entrenamiento, facilitando así la convergencia hacia una solución óptima. En segundo lugar, una función sinusoidal presenta otra propiedad matemática conveniente: el hecho de que su derivada también sea una función sinusoidal implica que calcular las derivadas (una operación esencial durante el entrenamiento) resulte computacionalmente eficiente. Esta eficiencia contribuye a agilizar el proceso de ajuste de los pesos y mejora la velocidad de entrenamiento [5].

Inicialización de los parámetros de la red

Para la inicialización de parámetros en SIREN, sus autores proponen una inicialización especial para los pesos y los sesgos [5]: para los sesgos, a partir de una distribución uniforme entre $[-\pi, \pi]$, y para los pesos, utilizando la siguiente fórmula:

$$W_{ij} \sim \mathcal{U}\left(-\sqrt{\frac{6}{n}}, \sqrt{\frac{6}{n}}\right), \quad (3.9)$$

donde n es el número de entradas a la neurona. Esto asegura que las activaciones de las neuronas estén distribuidas de manera uniforme en el intervalo $[-1, 1]$ para una entrada que siga una distribución $\mathcal{N}(0, 1)$.

Aplicaciones

SIREN destaca en la representación y la reconstrucción de imágenes bidimensionales y escenas tridimensionales. SIREN puede capturar la estructura detallada de las imágenes bidimensionales con gran fidelidad, manejando texturas finas y patrones complicados con gran precisión. En

escenas tridimensionales, SIREN es capaz de recrear geometrías complejas y heterogéneas. También puede manejar los campos de luz incidentes, lo que facilita la generación de escenas tridimensionales detalladas [5].

Más allá de las imágenes y las escenas, SIREN también se puede aplicar a la modelización de campos vectoriales, un componente importante en áreas como la física de fluidos o el electromagnetismo. En este contexto, SIREN se puede utilizar para representar la complejidad de estos campos con un alto grado de precisión [5].

3.2.2. Características de Fourier (*Fourier features*)

Las características de Fourier (*Fourier features*) [6], al igual que SIREN, tienen sus raíces en el análisis de Fourier [35]. Ambas se basan en el mismo principio fundamental de la descomposición de funciones en ondas sinusoidales. Sin embargo, mientras que SIREN integra la periodicidad sinusoidal en las capas de una red neuronal, las características de Fourier transforman las entradas a la red en un conjunto de características basadas en la frecuencia.

Mapeo de la entrada

Para realizar esta transformación se utiliza el mapeo de características de Fourier gaussiano [6]:

$$\gamma(\mathbf{v}) = [\cos(2\pi \mathbf{B}\mathbf{v}), \sin(2\pi \mathbf{B}\mathbf{v})]^T , \quad (3.10)$$

donde cada entrada en la matriz de mapeo $\mathbf{B} \in \mathbb{R}^{m \times d}$ se muestrea a partir de una distribución gaussiana $N(0, \sigma)$ y

\mathbf{v} es el vector de coordenadas espaciales normalizadas $\mathbf{v} \in \mathbb{R}^{d \times 1}$, en este caso, las posiciones de los píxeles en la imagen (en un rango de 0 a 1);

\mathbf{B} es la matriz de bases de Fourier $\mathbf{B} \in \mathbb{R}^{m \times d}$. Es una matriz de proyección que mapea las coordenadas espaciales a un espacio de características de Fourier, en nuestro caso $d=2$, y m es un hiperparámetro que depende de la resolución de las imágenes y de la complejidad del modelo;

$\gamma(\mathbf{v})$ es el vector de características de Fourier, que se obtiene aplicando las funciones coseno y seno a las coordenadas espaciales proyectadas a la dimensión $k = 2m$. Estas características se usarán como entrada de la red neuronal.

Consideramos el problema de regresión de imágenes 2D, en el que nos interesa predecir valores de píxeles para una imagen de salida basados en las coordenadas de los píxeles de una imagen de entrada. Cuando mapeamos esta entrada a la red mediante la matriz \mathbf{B} , cada coordenada se descompone en una suma de ondas sinusoidales de diferentes frecuencias. Estas nuevas características sinusoidales se usan como entrada a una red neuronal. La red, entonces, genera predicciones que se pueden interpretar como los valores de intensidad de los píxeles. De esta manera, el mapeo sinusoidal inicial permite que la red capte patrones complejos en las coordenadas de la imagen de entrada, lo que resulta en una regresión más precisa de la imagen de salida. Por último es importante dejar claro que estas características sinusoidales se calculan

directamente a partir de las coordenadas de entrada y no son parte del proceso de aprendizaje de la red.

Un aspecto destacable de las características de Fourier es su capacidad para mejorar el rendimiento de las redes neuronales sin alterar su arquitectura general. Este proceso implica un preprocesamiento de las entradas que aumenta su dimensionalidad, pero no requiere ninguna modificación en sus capas ocultas o de salida. Esto significa que las características de Fourier pueden integrarse fácilmente en cualquier arquitectura de red existente, y las convierte en una herramienta poderosa y flexible que permite a las redes neuronales manejar tareas de alta complejidad con mayor eficacia.

Aplicaciones

Las características de Fourier han demostrado ser útiles en varias tareas como la reconstrucción de imágenes, la representación y renderización de escenas tridimensionales, el aprendizaje de texturas de alta frecuencia y la predicción de iluminación [6]:

- **Reconstrucción de imágenes:** permite a las redes neuronales aprender y recrear imágenes con detalles finos y patrones periódicos complejos. Una red equipada con características de Fourier puede aprender la textura detallada de un objeto o la repetición periódica de patrones en una imagen, mejorando significativamente la calidad de la imagen recreada [6].
- **Representación y renderización de escenas tridimensionales:** mediante un enfoque de regresión de funciones, los autores demuestran que su método puede utilizarse para aprender a renderizar escenas tridimensionales a partir de entradas de baja dimensión (coordenadas bidimensionales o tridimensionales) [6].
- **Aprendizaje de texturas de alta frecuencia:** permite a las redes neuronales aprender a reproducir texturas de alta frecuencia, lo cual puede ser útil en una variedad de tareas de procesamiento de imágenes y gráficos [6].
- **Predicción de iluminación:** utilizando un conjunto de datos de iluminación del mundo real, las redes de este tipo pueden predecir la distribución de luz en una escena a partir de una entrada de baja dimensión (la dirección de la luz) [6].

3.3. Experimentos con las imágenes solares

Para nuestro primer objetivo, nos centraremos en replicar los experimentos presentados en los artículos de SIREN [5] y de las características de Fourier (*Fourier features*) [6] relativos a la regresión de imágenes bidimensionales utilizando nuestros propios datos. Como expusimos en el capítulo 1, debemos ajustar una red neuronal totalmente conectada de tipo *neural field* para recrear una porción de una de las 21 imágenes de nuestro conjunto de datos, y conseguir mapear las coordenadas de entrada a los valores de intensidad de cada píxel. En nuestro caso debemos encontrar la función:

$$\Phi : \mathbb{R}^2 \mapsto \mathbb{R}, \mathbf{x} \rightarrow \Phi(\mathbf{x}) , \quad (3.11)$$

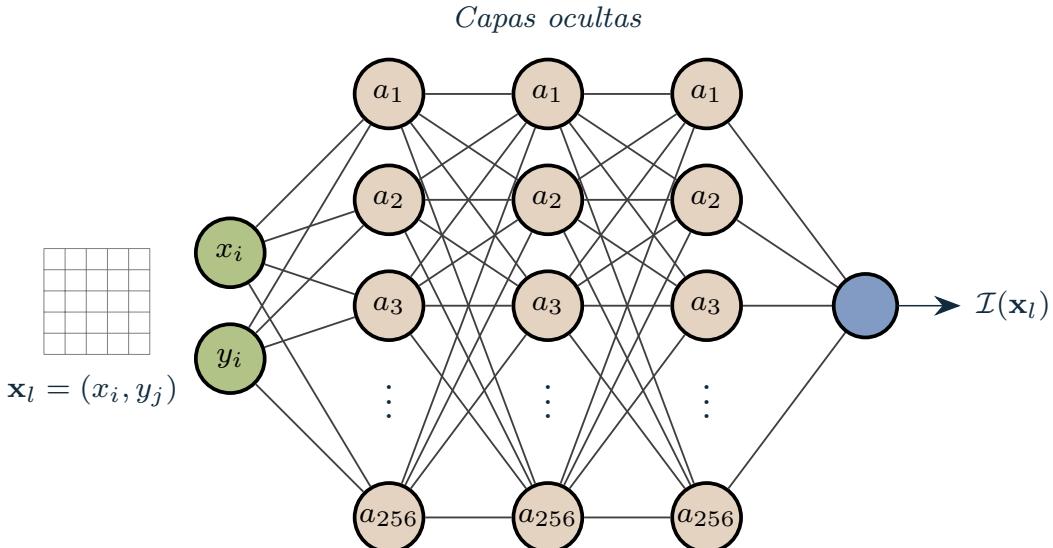


Figura 3.1: Configuración de la red neuronal para el experimento con SIREN. La entrada está formada por las coordenadas espaciales de la imagen normalizadas al intervalo [0, 1], consta de 3 capas ocultas de 256 neuronas y la capa de salida $\mathcal{I}(\mathbf{x}_l)$, que representa la intensidad del píxel para las coordenadas de entrada.

que parametrice una imagen discreta, \mathcal{I} , de manera continua. La imagen define un conjunto de datos $\mathcal{D} = \{(\mathbf{x}_l, \mathcal{I}(\mathbf{x}_l))\}_m$ de coordenadas de píxeles $\mathbf{x}_l = (x_i, y_j)$ asociadas con sus intensidades $\mathcal{I}(\mathbf{x}_l)$. Con esta transformación, todos los valores de x se mapearán al intervalo [0,1].

3.3.1. Regresión de imágenes bidimensionales con SIREN

La arquitectura que usamos en este experimento es una red neuronal de perceptrón multicapa (MLP) completamente conectada [4, 36], compuesta por una capa de entrada, que recibe las coordenadas espaciales, tres capas ocultas de 256 neuronas cada una y una capa de salida con una única neurona (Fig. 3.1), donde se realizará la comparación entre el valor de la intensidad del píxel obtenido por la red para las coordenadas de entrada y la etiqueta correspondiente con la intensidad real del píxel para esas coordenadas. Escalamos tanto la entrada al modelo (coordenadas de los píxeles de la imagen) como las etiquetas de los datos (las intensidades de cada píxel) a valores uniformemente distribuidos en el intervalo [0, 1]. Esta técnica se utiliza ampliamente en el aprendizaje automático para mejorar la convergencia y la eficacia de los algoritmos. Cuando diferentes variables tienen rangos variados, los algoritmos, especialmente aquellos basados en gradientes, pueden tardar más en converger o pueden ser guiados erróneamente hacia mínimos locales no óptimos. Escalar las variables al mismo rango mitiga estos problemas y mejora la eficacia del entrenamiento. Esto optimiza el aprendizaje de la red, facilita una representación adecuada y asegura una propagación eficiente del gradiente a lo largo de sus capas.

Como adelantamos en el capítulo 1, utilizaremos un fragmento de la imagen original para llevar a cabo la reconstrucción. Esto permite un procesamiento más eficiente [14] y resulta particularmente útil al utilizar hardware con limitaciones de memoria, como las GPU [37, 38]. El tamaño que hemos elegido para el fragmento de la imagen es de 512×512 píxeles. Para el conjunto de datos de entrenamiento, hemos tomado una submuestra de la imagen compuesta por cada píxel de subíndice impar de ambas dimensiones (altura y ancho, con indexación desde

Parámetro	Valor
Normalización de las coordenadas	[0, 1]
Normalización de los píxeles	[0, 1]
Tamaño del recorte de imagen	512 x 512 píxeles
División de los datos	50 % entrenamiento, 100 % prueba
Inicialización de pesos (primera capa)	$\mathcal{U}[-\frac{1}{n}, \frac{1}{n}]$
Inicialización de pesos (capas ocultas y de salida)	$\mathcal{U}[-\sqrt{\frac{6}{n}}/\omega_0, \sqrt{\frac{6}{n}}/\omega_0]$
Inicialización de sesgos	0
Parámetro de frecuencia (ω_0)	10, 30, 50, 100
Parámetro de frecuencia (ω)	10, 30, 50, 100
Capas ocultas	3
Neuronas en las capas ocultas	256
Optimizador	<i>Adam</i>
Tasa de aprendizaje (α)	10^{-4}
Número de épocas	2000
Métricas	MSE, PSNR

Tabla 3.1: Resumen de los parámetros utilizados en el experimento de regresión de una imagen bidimensional usando SIREN.

0). Evaluaremos el modelo durante el entrenamiento utilizando como conjunto de pruebas la totalidad de los píxeles de la imagen.

Siguiendo la metodología de la publicación original, los parámetros de la red se ajustaron de la siguiente manera: para inicializar los pesos de la primera capa utilizamos una distribución uniforme en el rango $[-\frac{1}{n}, \frac{1}{n}]$, donde n es el número de entradas a la capa. Para las capas ocultas y de salida, una distribución uniforme en el rango $[-\sqrt{\frac{6}{n}}/\omega_0, \sqrt{\frac{6}{n}}/\omega_0]$. Los sesgos de todas las capas se inicializaron con el valor $b_i = 0$. Además, con el objetivo de encontrar el parámetro de frecuencia ω_0 de la función de activación sinusoidal adecuado para nuestro conjunto de datos, entrenamos la red con diferentes valores (10, 30, 50, 100). Este parámetro se mantiene constante durante el entrenamiento y es crucial para la convergencia del modelo.

El entrenamiento de la red se llevó a cabo durante 2000 ciclos (*epochs*) utilizando el algoritmo de optimización *Adam* [26] con una tasa de aprendizaje (Ec. 2.15 y 2.16) de 10^{-4} . La métrica que hemos utilizado para evaluar el rendimiento de la red es la relación señal-ruido (PSNR) que se calcula como:

$$\text{PSNR} = 10 \cdot \log_{10} \left(\frac{\text{MAX}_I^2}{\text{MSE}} \right), \quad (3.12)$$

donde MAX_I es el valor máximo posible del píxel en la imagen (1 en nuestro experimento) y MSE es el error cuadrático medio. Esta métrica es especialmente útil para comparar la fidelidad de las imágenes reconstruidas. Al emplear una escala logarítmica, da más importancia a los errores que son perceptualmente más relevantes. Por otro lado, al considerar MAX_I , la PSNR proporciona un contexto que permite juzgar si un MSE dado es alto o bajo en relación al rango de la imagen.

La Tabla 3.1 resume los valores de normalización de los datos, su tamaño y distribución, la

inicialización de los pesos, sesgos y la frecuencia de la función de activación del modelo, así como los parámetros y métricas usados durante la fase de entrenamiento.

PSNR	$\omega_0 = 10$	$\omega_0 = 30$	$\omega_0 = 50$	$\omega_0 = 100$
Entrenamiento	29.51	37.41	39.66	49.13
Prueba	29.49	37.34	38.50	38.39

Tabla 3.2: Valores de la PSNR para los datos de entrenamiento y prueba en función de diferentes valores de ω_0 en el experimento de regresión de imágenes bidimensionales con SIREN.

Los resultados en la Tabla 3.2 indican cómo se comporta el modelo SIREN en la tarea de regresión de imágenes bidimensionales. A medida que el valor de ω_0 aumenta, la PSNR para los datos de entrenamiento también tiende a aumentar. Sin embargo, para los datos de prueba, la PSNR alcanza un pico alrededor de $\omega_0 = 50$ y luego disminuye ligeramente para $\omega_0 = 100$. Esto sugiere que, aunque el modelo puede capturar con precisión las intensidades de los píxeles en el conjunto de entrenamiento para valores más altos de ω_0 , puede no generalizar tan bien en datos no vistos, posiblemente debido a un sobreajuste.

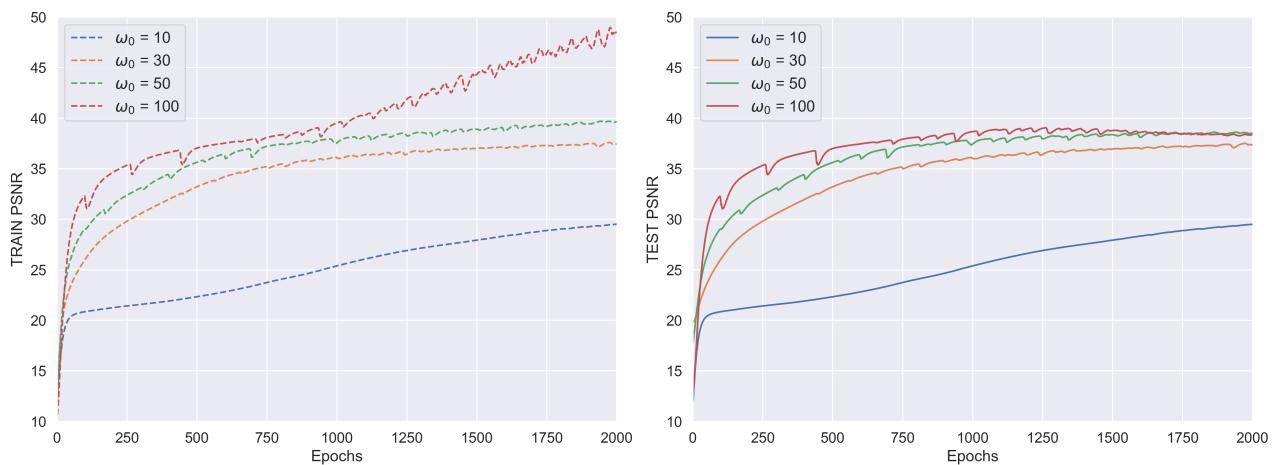


Figura 3.2: Evolución de la PSNR para los datos de entrenamiento (*train*) y prueba (*test*) con diferentes valores de ω a lo largo de las 2000 épocas (*epochs*).

En la Fig. 3.2 se ve claramente la evolución de la PSNR a lo largo de las épocas para diferentes valores de ω . En el gráfico de entrenamiento (izquierda), se observa que la PSNR para todos los valores de ω tiende a aumentar con el número de épocas, corroborando lo que se mencionó anteriormente sobre cómo el modelo se ajusta mejor a los datos de entrenamiento con valores más altos de ω . En particular, el valor de $\omega = 100$ muestra un crecimiento continuo, alcanzando los valores más altos de la PSNR. Por otro lado, en el gráfico de prueba (derecha), aunque todos los valores de ω muestran un aumento inicial de la PSNR, hay variaciones significativas en su comportamiento a medida que avanzan las épocas. El $\omega = 50$ parece tener el mejor rendimiento, mostrando un crecimiento sostenido y alcanzando un pico antes de estabilizarse, mientras que $\omega = 100$, después de un rápido crecimiento inicial, tiene un rendimiento más bajo en comparación con los otros valores a medida que avanza el número de épocas. Al observar más de cerca el gráfico de prueba, parece que el sobreajuste para $\omega = 100$ comienza aproximadamente después de las 1000 épocas, ya que la PSNR comienza a estabilizarse y luego disminuye ligeramente. Esto respalda la observación anterior sobre el potencial sobreajuste con un ω muy alto.

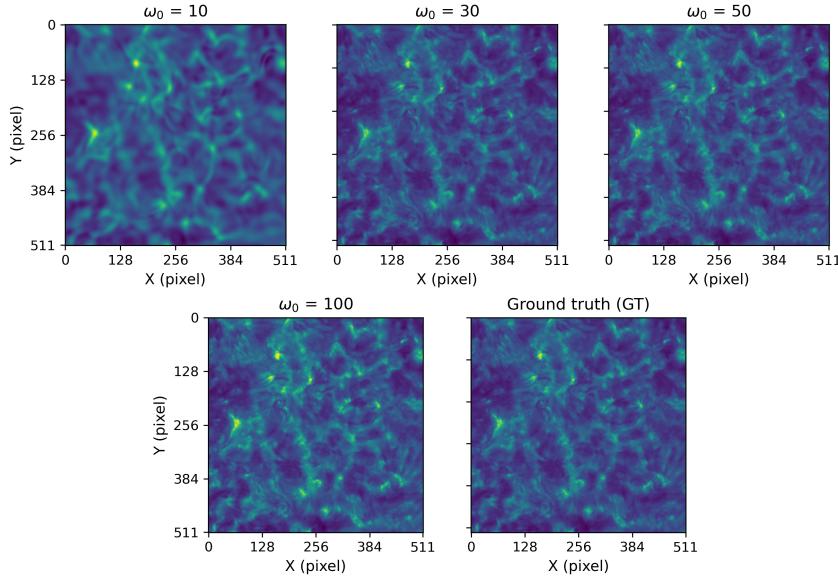


Figura 3.3: Reconstrucción de una porción de 512×512 píxeles de una imagen del conjunto de datos utilizando la arquitectura SIREN. Las imágenes representan reconstrucciones con el conjunto de datos de prueba para diferentes valores de ω_0 y la imagen original (Ground truth, GT).

Por lo general, se considera que la reconstrucción de una imagen es de alta calidad cuando el valor de la PSNR es de alrededor de 30 o superior [7]. Los valores altos, tanto en los datos de entrenamiento como en los de prueba, indican que las imágenes重建adas por la red están cercanas a las imágenes originales. La calidad de las reconstrucciones visualmente mostrada en la Fig. 3.3 es consistente con estos valores altos de la PSNR. Las imágenes重建adas que exceden este umbral de PSNR suelen tener una fidelidad visual que se asemeja estrechamente a las imágenes originales, lo que confirma la capacidad de la red para producir reconstrucciones de alta calidad.

3.3.2. Regresión de imágenes bidimensionales con características de Fourier (*Fourier features*)

Para el experimento de regresión con las características de Fourier también utilizamos una red neuronal de tipo perceptrón multicapa (MLP) completamente conectada [4, 36]. Esta red consta de una capa que mapea las coordenadas de entrada a una primera capa de $k = 256$ neuronas, tres capas ocultas más y una capa de salida. En las tres capas ocultas disponemos 256 neuronas que usan una función de activación ReLU (Fig. 2.2d). La capa de salida, que es responsable de la generación de la intensidad de los píxeles de la imagen, consta de una única neurona y utiliza una función de activación sigmoidal (Fig. 2.2b) para asegurar que los valores generados estén en el rango de 0 a 1.

Preparamos los datos de entrada siguiendo el mismo procedimiento que el artículo original [6]: en primer lugar fragmentamos las imágenes al tamaño de 512×512 píxeles y normalizamos el valor de sus intensidades entre 0 y 1. A continuación, creamos la malla unitaria, mapeando todas las coordenadas a valores dentro del intervalo $[0, 1]$ [23]. Esta estrategia resulta especialmente efectiva con la función de activación ReLU (Fig. 2.2d), ya que asegura que todas las salidas de la función son no negativas, evitando así el truncamiento a cero y permitiendo a las neuronas producir respuestas no nulas [39]. Esto, a su vez, puede mejorar la capacidad de la red para

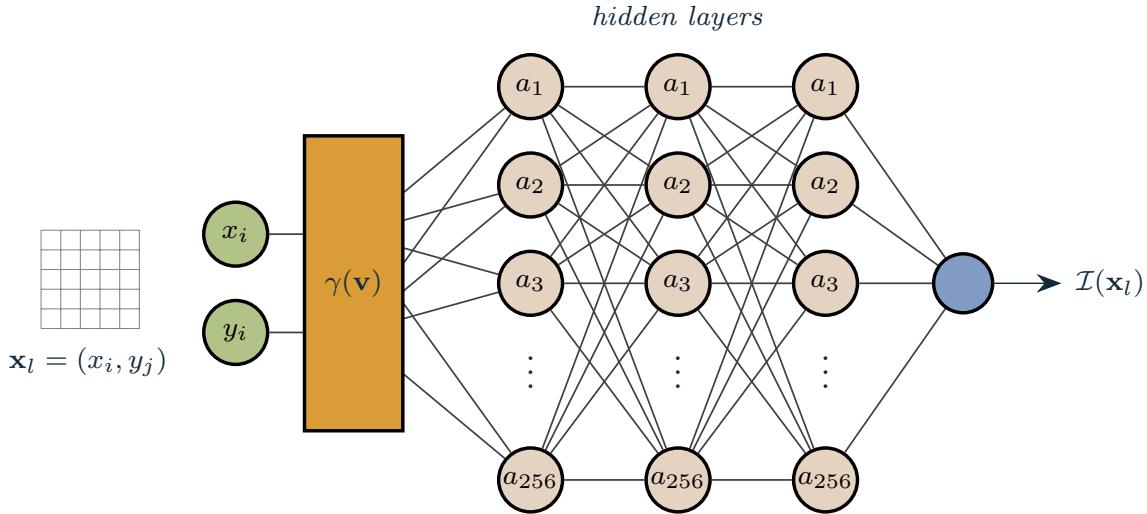


Figura 3.4: Configuración de la red neuronal para el experimento de características de Fourier. La entrada está formada por las coordenadas espaciales de la imagen $\mathbf{x}_l = (x_i, y_j)$ normalizadas a la rejilla unidad en el intervalo $[0,1]$, que posteriormente se mapean para obtener el vector de características de Fourier (Ec. 3.10) con k neuronas (en este caso 256). La red tiene 3 capas ocultas con 256 neuronas cada una. Por último, la capa de salida representa la intensidad del píxel aprendida.

aprender y capturar características significativas de los datos [40]. Además, mantener todas las entradas en una escala similar es importante, ya que puede ayudar a evitar problemas de estabilidad durante el entrenamiento, lo que conduce a un mejor rendimiento y precisión del modelo.

Antes de alimentar la red, las coordenadas de la imagen se someten a una transformación utilizando el mapeo de características de Fourier [6]. Para apreciar el impacto de la aplicación de las características de Fourier en el rendimiento de la red, comparamos las siguientes mapeos del vector de características $\gamma(\mathbf{v})$:

- Sin mapeo: $\gamma(\mathbf{v}) = \mathbf{v}$.
- Mapeo básico: $\gamma(\mathbf{v}) = [\cos(2\pi\mathbf{v}), \sin(2\pi\mathbf{v})]^T$.
- Mapeo de características de Fourier gaussiano: $\gamma(\mathbf{v}) = [\cos(2\pi\mathbf{B}\mathbf{v}), \sin(2\pi\mathbf{B}\mathbf{v})]^T$, donde cada entrada en $\mathbf{B} \in \mathbb{R}^{m \times d}$ se selecciona de $N(0, \sigma)$ y $k = 2m$ es el número de características de Fourier (acorde con el experimento de referencia [6], utilizaremos $k = 256$). Como mencionamos en el apartado 3.2.2, la matriz de mapeo \mathbf{B} se inicializa una vez al comienzo del experimento y las características calculadas a partir de ella al aplicarlas a las coordenadas de la imagen son las que alimentan la red y no se actualizan durante el proceso de entrenamiento.

Con el fin de comparar ambos modelos entrenamos la red durante 2000 épocas utilizando el algoritmo de optimización *Adam* [26] y una tasa de aprendizaje (Ec. 2.15 y 2.16) de 10^{-4} . De la misma manera que con SIREN, las métricas que hemos utilizado para evaluar el rendimiento de la red son el MSE y la PSNR. La Tabla 3.3 resume los valores del experimento.

En los resultados presentados en la Tabla 3.4 se observa un patrón interesante del valor de σ . Cuando se utiliza un valor de $\sigma = 1$, se logra una PSNR bastante aceptable tanto con los conjuntos de entrenamiento como de prueba, lo que indica que el modelo puede aprender y

Parámetro	Valor
Normalización de las coordenadas	[0, 1]
Normalización de los píxeles	[0, 1]
Tamaño del recorte de imagen	512 x 512 píxeles
Transformación de las coordenadas	Mapeo de características de Fourier
Mapeos de características $\gamma(\mathbf{v})$	Sin mapeo, mapeo básico, Fourier gaussiano
Número de características de Fourier (k)	256
\mathbf{B}	$\in \mathbb{R}^{128 \times 2}$
Desviación estándar en Fourier gaussiano	$\sigma = 1$, $\sigma = 10$ (óptimo), $\sigma = 100$ (sobreajuste)
Capas ocultas	3
Neuronas en la primera capa	k
Neuronas las capas ocultas	256
Optimizador	<i>Adam</i>
Tasa de aprendizaje (α)	10^{-4}
Número de épocas	2000
Métricas	MSE, PSNR

Tabla 3.3: Resumen de los parámetros utilizados en el experimento de regresión de una imagen bidimensional usando características de Fourier (*Fourier features*).

generalizar razonablemente bien. Al aumentar σ a 10 se logra una mejora en la PSNR en ambos conjuntos, con unos valores de 40.76 en el conjunto de entrenamiento y 39.02 en el conjunto de prueba. Esto demuestra que el uso de un valor de σ más grande permite al modelo aprender patrones más complejos y detallados en la imagen, lo que mejora significativamente la calidad de la reconstrucción.

PSNR	Mapeo gaussiano				
	Sin mapeo	Mapeo simple	$\sigma = 1$	$\sigma = 10$	$\sigma = 100$
Train	21.55	26.05	29.92	39.63	36.03
Test	21.55	26.03	30.09	38.22	17.85

Tabla 3.4: Resultados de la relación señal-ruido (PSNR) en la tarea de regresión de la imagen bidimensional para los conjuntos de datos de entrenamiento y prueba, con los diferentes mapeos de las características de Fourier.

No obstante, al incrementar la desviación estándar a $\sigma = 100$, el modelo parece sufrir de sobreajuste. Aunque el rendimiento del modelo con respecto a los datos de entrenamiento sigue mejorando (PSNR = 36.03), la PSNR para los datos de prueba disminuye a 17.85. Esto indica que, a pesar de que el modelo puede aprender los datos de entrenamiento con un nivel extremo de detalle, su capacidad para generalizar a nuevos datos se ve seriamente obstaculizada. En contraposición, sin mapeo o con un mapeo simple, el modelo es incapaz de capturar adecuadamente la complejidad de la imagen, lo que resulta en una reconstrucción de calidad inferior y una PSNR disminuida (Fig. 3.5).

En términos generales, estos resultados se pueden contrastar visualmente con la calidad de las reconstrucciones para cada uno de los mapeos mostrados en la Fig. 3.6, donde queda patente la importancia de seleccionar y ajustar cuidadosamente los parámetros de las características de Fourier cuando se aplican a tareas de regresión de imágenes.

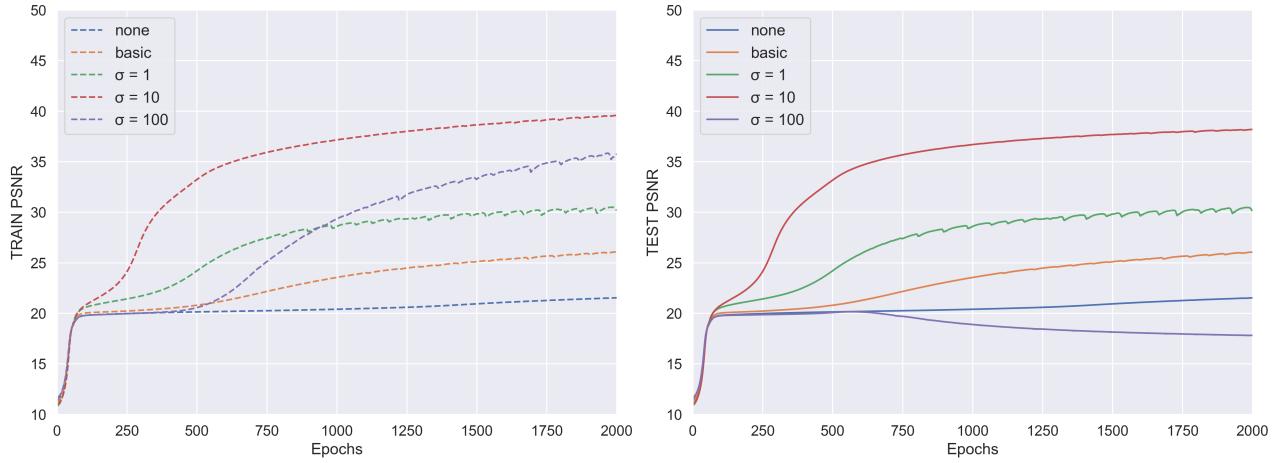


Figura 3.5: Evolución de la relación señal-ruido (PSNR) a lo largo de las 2000 épocas de entrenamiento de la red para los conjuntos de entrenamiento (*train*) y prueba (*test*). Cada una de las líneas corresponde al mapeo utilizado en el entrenamiento: Sin mapeo (*none*): $\gamma(\mathbf{v}) = \mathbf{v}$, mapeo básico (*basic*): $\gamma(\mathbf{v}) = [\cos(2\pi\mathbf{v}), \sin(2\pi\mathbf{v})]^T$ y mapeo de características de Fourier gaussiano: $\gamma(\mathbf{v}) = [\cos(2\pi\mathbf{B}\mathbf{v}), \sin(2\pi\mathbf{B}\mathbf{v})]^T$, con $\sigma = 1$, $\sigma = 10$ y $\sigma = 100$.

3.4. Discusión de los resultados

En este capítulo hemos demostrado que las redes basadas en campos neuronales implícitos que usan funciones de activación sinusoidales (SIREN) [5] o características de Fourier (*Fourier features*) [6] ofrecen enfoques reales y muy eficaces para la tarea de regresión de imágenes. En particular, estos métodos han demostrado su capacidad para aprender y reproducir con precisión patrones complejos y de alta frecuencia, tal como evidencian los altos valores de la PSNR obtenidos en los experimentos realizados en el apartado 3.3.

Aunque en los experimentos originales de SIREN sus autores utilizaron un único conjunto de datos para el entrenamiento [5], nuestra adaptación del método demostró resultados alentadores al dividir los datos en conjuntos de entrenamiento y prueba (Fig. 3.2). Esto resalta la robustez de SIREN, incluso cuando se intenta generalizar a partir de datos vistos a datos no vistos, un criterio fundamental para nuestra aplicación específica.

El uso de las características de Fourier (*Fourier features*) también proporcionó resultados prometedores (Fig. 3.5). Cuando se ajustan adecuadamente, permiten que el modelo aprenda patrones y estructuras complejas de los datos de manera efectiva y eficiente. Además, los resultados también revelaron la importancia de seleccionar cuidadosamente los parámetros, en particular, el valor de la desviación estándar σ , para evitar el sobreajuste y maximizar la capacidad de generalización del modelo.

Los resultados obtenidos tanto con las características de Fourier como con SIREN son prometedores en términos de la PSNR obtenida (Fig. 3.7), lo que sugiere que ambas pueden ser técnicas adecuadas para nuestro objetivo de la superresolución espectral en líneas de absorción solares. Aunque la regresión de imágenes puede parecer una tarea concreta y específica, la generalización ha sido crucial en nuestro escenario. Buscamos la capacidad de generar imágenes intermedias en el cubo de datos solares a partir de las imágenes aprendidas. Es decir, el modelo debe ser capaz de generar una nueva imagen que no está presente en los datos de entrenamiento. Por lo tanto, una evaluación rigurosa del rendimiento de los modelos en los conjuntos de entrenamiento y

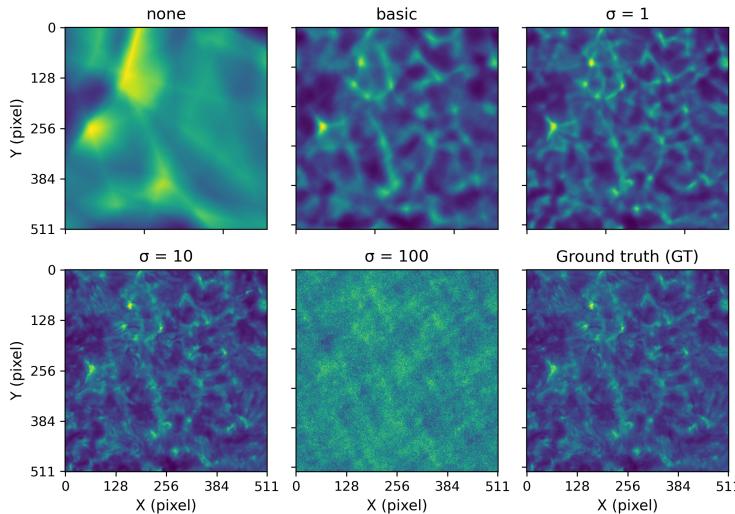


Figura 3.6: Resultado de la regresión de una porción de 512×512 píxeles de una las imágenes de nuestro conjunto de datos usando características de Fourier. Se muestran las imágenes reconstruidas usando los datos de prueba y diferente mapeos: Sin mapeo (*none*): $\gamma(\mathbf{v}) = \mathbf{v}$, mapeo básico (*basic*): $\gamma(\mathbf{v}) = [\cos(2\pi\mathbf{v}), \sin(2\pi\mathbf{v})]^T$ y mapeo de características de Fourier gaussiano: $\gamma(\mathbf{v}) = [\cos(2\pi\mathbf{B}\mathbf{v}), \sin(2\pi\mathbf{B}\mathbf{v})]^T$, con $\sigma = 1$, $\sigma = 10$ y $\sigma = 100$. Además se muestra la imagen original o datos verdaderos (*Ground truth, GT*).

prueba es esencial.

Para finalizar, teniendo en cuenta los resultados discutidos en este capítulo y basándonos en la calidad de las imágenes reproducidas por los modelos (Fig. 3.8), ambas técnicas se presentan como una opción viable y atractiva para avanzar hacia la superresolución espectral, dadas su flexibilidad para adaptarse a cualquiera de las arquitecturas existentes y su capacidad para capturar patrones complejos en los datos.

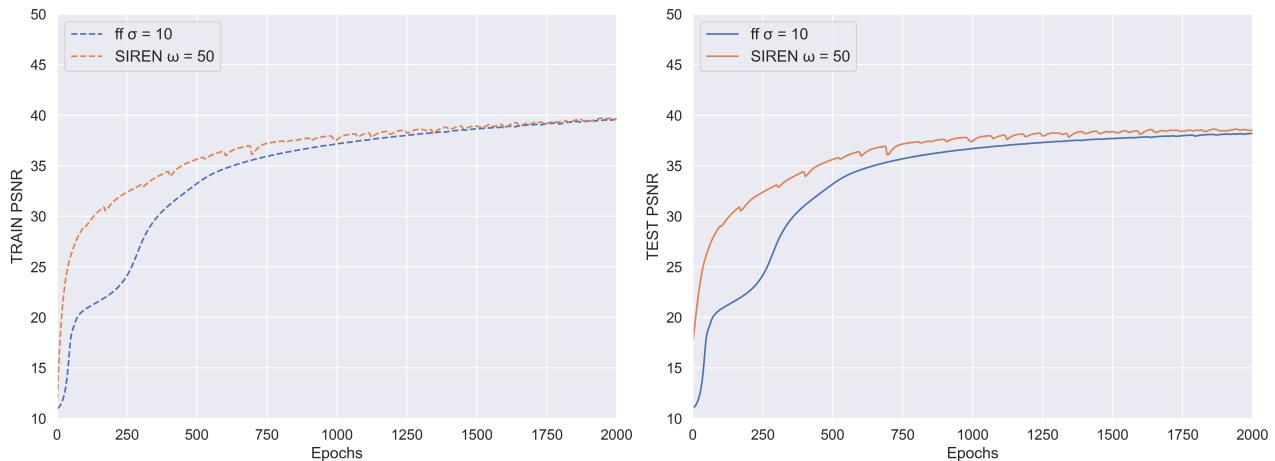


Figura 3.7: Evolución de la relación señal-ruido (PSNR) a lo largo de las 2000 épocas de entrenamiento de la red para los conjuntos de entrenamiento (*train*) y prueba (*test*) que obtuvieron mejor rendimiento (FF = Fourier features).

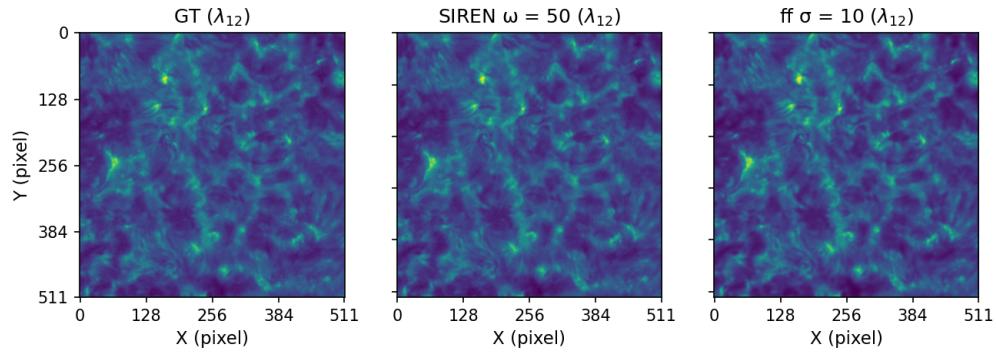


Figura 3.8: Resultado de la regresión de una porción de 512×512 píxeles de una las imágenes de nuestro conjunto de datos usando SIREN y características de Fourier (FF). Además se muestra la imagen original o datos verdaderos (*Ground truth, GT*).

Capítulo 4

Hacia la superresolución espectral

En el capítulo anterior demostramos que las redes basadas en campos neuronales implícitos, utilizando funciones de activación sinusoidal (SIREN) [5] o características de Fourier [6], son efectivas para la tarea de la regresión de imágenes pseudomonocromáticas de la superficie solar. Los experimentos detallados revelaron que estas técnicas pueden aprender y reproducir patrones de alta frecuencia con un alto grado de complejidad. Es más, argumentamos que ambas son una opción efectiva y una base sólida en la que fundamentar nuestro avance en el desafío de la superresolución espectral en líneas de absorción del espectro del Sol.

En este capítulo introduciremos el eje de longitud de onda a nuestra serie de imágenes pseudomonocromáticas, creando un cubo de datos tridimensional (Fig. 4.2) que permita la interpolación espectral continua. Para ello empezaremos prolongando el experimento descrito en el apartado 3.3.2 con nuestro nuevo conjunto de datos, con el objetivo de generar imágenes intermedias en el cubo de datos solares a partir de las imágenes aprendidas. Proporcionaremos una visión detallada de este proceso, examinando la preparación de los datos y la selección de la arquitectura de red más adecuada para nuestra tarea.

4.1. El conjunto de datos

Como vimos en el capítulo 1, disponemos de un cubo de datos espetrales reales $I(x, y, \lambda)$ de la línea fotosférica de absorción del hierro neutro centrada en 6302 Å (Fe I λ 6302). Este conjunto consta de 21 imágenes pseudomonocromáticas, cada una con dimensiones de 966×964 píxeles, correspondientes a un área extensa de la superficie solar (Fig. 4.1). Dicho de otra manera, estas imágenes representan un muestreo espectral de la línea de absorción de Fe I λ 6302 en 21 segmentos de longitud de onda específicos.

Intuitivamente, podemos visualizar este conjunto de datos como un cubo tridimensional donde la dimensión x y la dimensión y corresponden a los píxeles en la imagen, y la dimensión λ representa las diferentes longitudes de onda en la línea de absorción de Fe I λ 6302. Las 21 imágenes están equiespaciadas en el eje de λ , formando una pila o un cubo de imágenes en tres dimensiones. La Fig. 4.2 muestra una representación gráfica de este cubo, donde cada capa corresponde a una imagen etiquetada con un valor específico de λ_t con t de 1 a 21. Esta visualización en 3D permite una comprensión más clara de cómo la línea de absorción se muestrea en diferentes longitudes de onda, revelando características y patrones que pueden no ser evidentes cuando se examinan las imágenes individualmente. La elección de etiquetar las imágenes de 1 a 21 se hace por conveniencia y ayuda a conceptualizar la relación espacial y espectral de las imágenes dentro del cubo.

En la Fig. 4.3 se puede apreciar la variación de la intensidad media de los píxeles de cada una de las 21 imágenes, lo que muestra el perfil distintivo de la línea de absorción, con un mínimo de la intensidad media en su centro.

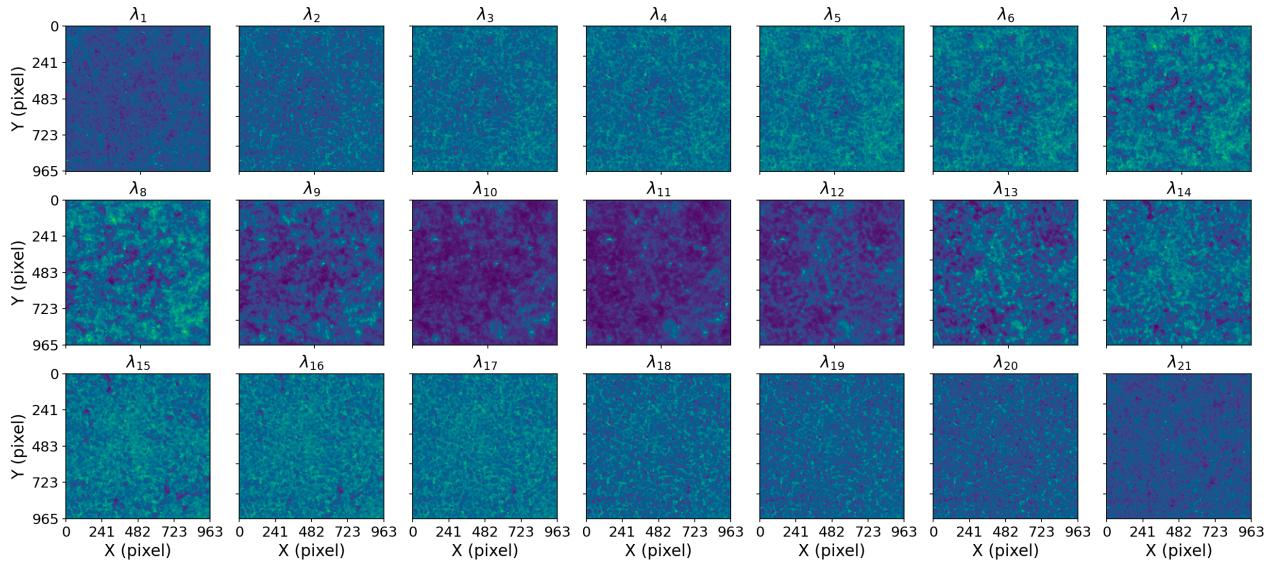


Figura 4.1: 21 imágenes que escanean la línea fotosférica de absorción del hierro neutro centrada en 6302 Å (FeI λ 6302). Cada imagen se ha etiquetado como λ_t con t de 1 a 21.

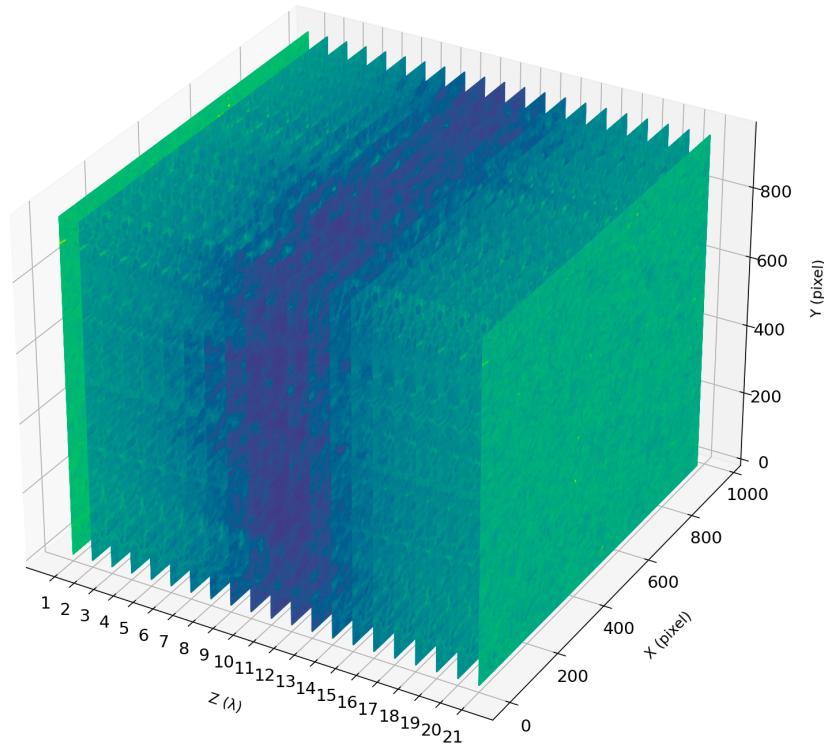


Figura 4.2: Representación tridimensional del cubo de imágenes que escanean la línea fotosférica de absorción de hierro neutro centrada en 6302 Å (FeI λ 6302). Las 21 imágenes están equiespaciadas en longitud de onda y etiquetadas como λ_t con t de 1 a 21.

Este perfil de absorción no solo subraya el carácter tridimensional del conjunto de datos, sino que también proporciona una perspectiva esencial de cómo cambian las estructuras observadas en la región del Sol estudiada con la longitud de onda dentro de la línea de absorción de FeI λ 6302. Esta información es vital para entender las propiedades físicas de esta región y

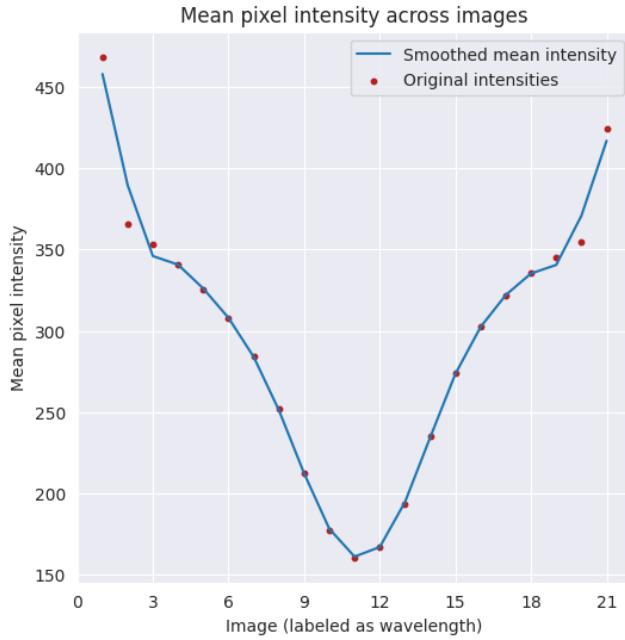


Figura 4.3: Perfil de la línea de absorción de Fe I $\lambda 6302$, que muestra la variación de la intensidad media del píxel a través de las 21 imágenes equiespaciadas. Las intensidades originales de los píxeles se muestran con puntos rojos (*original intensities*) y en azul una regresión suavizada del cambio de intensidad media a través de la linea de absorción (*smoothed mean intensity*)

constituye una herramienta valiosa para la interpretación y el análisis de la línea de absorción fotosférica del hierro neutro. Recordemos que en una línea espectral el cambio de intensidad de la absorción en función de la longitud de onda tiene que ver con la profundidad en la fotosfera solar (ver capítulo 1).

4.2. Reconstrucción del cubo de imágenes solares

Con nuestro nuevo enfoque buscamos no solo mapear las coordenadas espaciales x e y de la imagen, sino también incorporar el eje de la longitud de onda λ y que esto nos permita encontrar una función

$$\Phi : \mathbb{R}^3 \mapsto \mathbb{R}, (\mathbf{x}, \lambda) \rightarrow \Phi(\mathbf{x}, \lambda) \quad (4.1)$$

que parametrice nuestro cubo de datos de manera continua. La imagen tridimensional define un conjunto de datos $\mathcal{D} = \{(\mathbf{x}_l, \lambda_t, \mathcal{I}(\mathbf{x}_l, \lambda_t))\}_m$ de coordenadas de píxeles $\mathbf{x}_l = (x_i, y_j)$ y longitudes de onda λ_t asociadas con sus intensidades $\mathcal{I}(\mathbf{x}_l, \lambda_t)$. Utilizando la misma aproximación de *neural fields* que en el experimento del apartado 3.3.2, ahora debemos analizar cómo esta nueva dimensión influye en nuestra función de mapeo.

La forma en que alimentamos la red con los datos es esencial para su rendimiento. Ahora estamos trabajando con un cubo de datos tridimensional, por lo que cada entrada consistirá en las coordenadas (x_i, y_j, λ_t) . La estructura de estos datos requiere una manipulación previa para asegurar que la red pueda aprender los patrones tanto espaciales como espectrales dentro del cubo. En primer lugar, siguiendo con el procedimiento presentado en el apartado 3.3.2, recortamos cada una de las imágenes a un tamaño uniforme. En segundo lugar, normalizamos las intensidades de los píxeles al intervalo $[0, 1]$ con relación al máximo y mínimo de cada imagen

para garantizar que todos los valores estén en la misma escala. Esto facilita la convergencia durante el entrenamiento y mejora la estabilidad y el rendimiento del modelo como discutimos en el apartado 3.3.2. Finalmente, creamos una malla tridimensional que mapee todas las coordenadas a valores dentro del intervalo $[0, 1]$. Lo hacemos generando puntos equiespaciados en cada dimensión (x_i, y_j, λ_t) y combinándolos en una rejilla tridimensional (Fig. 4.4).

4.2.1. Experimentos con SIREN

La arquitectura SIREN que usamos en nuestra implementación se muestra en la Fig. 4.4. Consta de una capa de entrada que toma las coordenadas de píxeles (x_i, y_j, λ_t) y les aplica una función de activación sinusoidal con un parámetro de frecuencia ω_0 , 3 capas ocultas que también utilizan funciones de activación sinusoidales, cada una con su propio parámetro de frecuencia ω , y una capa de salida que produce el valor de intensidad del píxel para las coordenadas dadas.

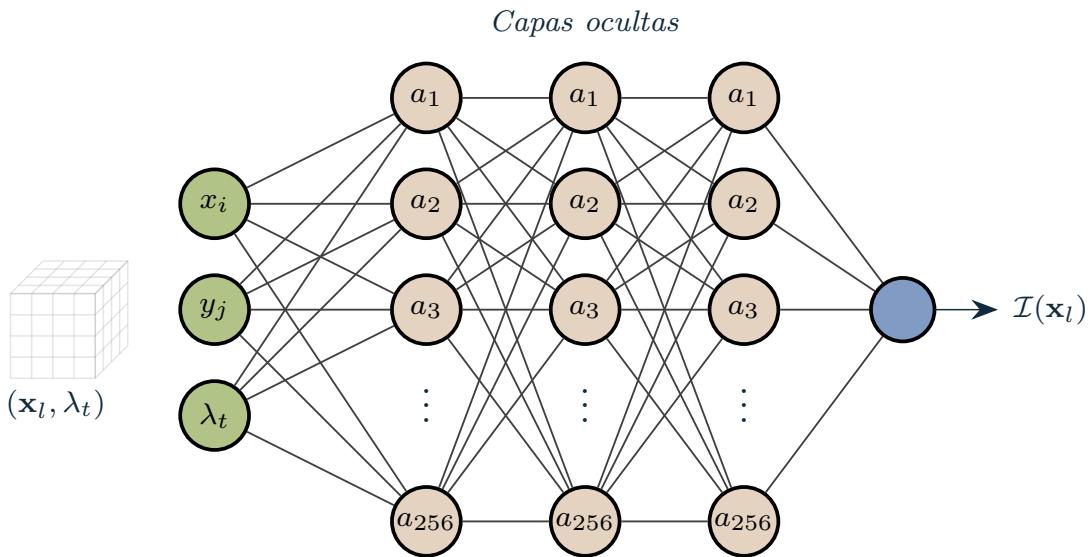


Figura 4.4: Configuración de la red neuronal para el experimento con SIREN. La entrada está formada por las coordenadas espaciales y de longitud de onda de cada imagen $(\mathbf{x}_l, \lambda_t)$ normalizadas a la rejilla unidad tridimensional, consta de 3 capas ocultas de 256 neuronas y la capa de salida $I(\mathbf{x}_l)$, que representa la intensidad del píxel para las coordenadas de entrada.

Exploramos diferentes parámetros de frecuencia ω_0 para la capa de entrada y ω para las capas ocultas. Concretamente, probamos los valores con los que obtuvimos los mejores resultados en el experimento del apartado 3.3.1, es decir $\omega_0 = 30$ y $\omega_0 = 50$ para la frecuencia de la capa de entrada y $\omega = 30$ y $\omega = 50$ para la frecuencia de las capas ocultas. La red fue entrenada a lo largo de 2000 *epochs* utilizando *Adam* como algoritmo de optimización [26], con una tasa de aprendizaje establecida en 10^{-4} (Ec. 2.15 y 2.16). Por último, evaluamos el rendimiento empleando la métrica de la relación señal-ruido (PSNR).

La Tabla 4.1 muestra que los valores de la PSNR obtenidos con los datos de prueba superan el umbral de 30 tanto para $\omega_0 = 30$ como para $\omega_0 = 50$. Estos valores sugieren que la red es capaz de aproximar las imágenes originales con un alto grado de precisión [7], lo que es crucial para abordar la superresolución espectral.

La reconstrucción efectiva de las imágenes es más evidente en las representaciones visuales. Como se muestra en la Fig. 4.5, las imágenes generadas por la red (parte superior de cada

PSNR	$\omega_0 = 30$	$\omega_0 = 50$
Entrenamiento	33.57	34.83
Prueba	33.72	35.02

Tabla 4.1: Valores de la PSNR para los datos de entrenamiento y prueba en función de diferentes valores de ω_0 en el experimento de regresión de las 21 imágenes pseudomonocromáticas con SIREN.

subfigura) presentan una notable similitud con las imágenes originales (parte inferior de cada subfigura), destacando la capacidad de la red para interpolar espectralmente dentro del cubo de datos.

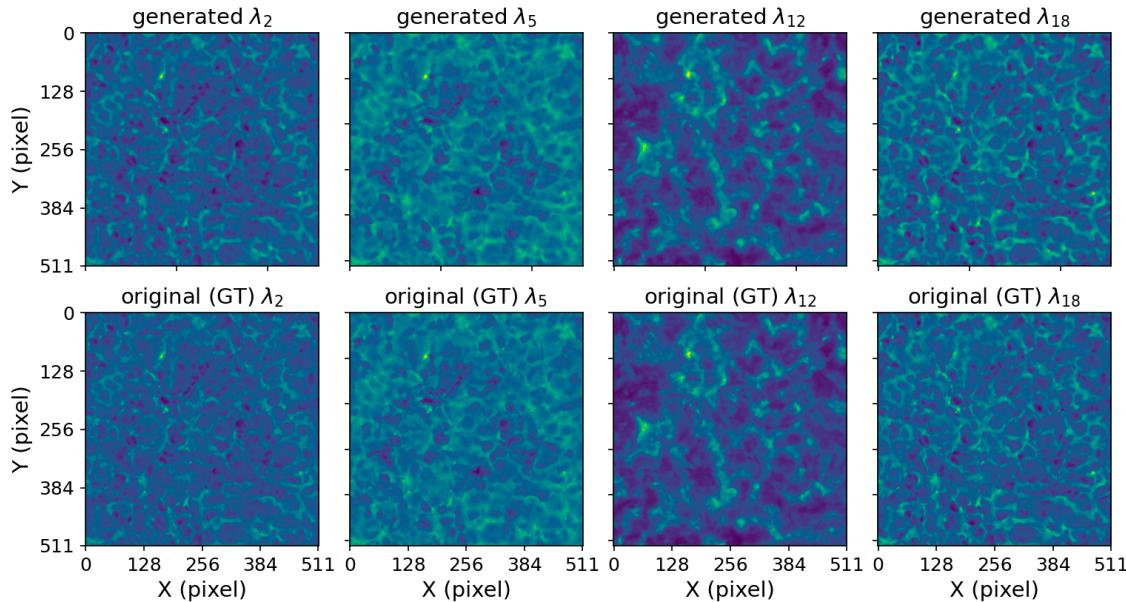


Figura 4.5: Reconstrucción de una porción de 512×512 píxeles de cuatro imágenes del conjunto de datos utilizando la arquitectura SIREN. Las imágenes superiores representan las reconstrucciones utilizando un valor de $\omega_0 = 50$ y las inferiores las originales (Ground truth, GT).

La evolución de la PSNR durante las fases de entrenamiento y prueba, representada en la Fig. 4.6, refleja la estabilidad y la consistencia en el rendimiento del modelo con un comportamiento que favorece la generalización. Esto es particularmente relevante cuando consideramos la interpolación espectral continua en nuestro cubo de datos 3D, un desafío clave en la reconstrucción de imágenes en la que la precisión es crucial.

4.2.2. Experimentos con *Fourier features*

Utilizaremos la misma arquitectura de red que en el apartado 3.3.2 para comprobar el rendimiento del *neural field* con nuestro nuevo conjunto de datos (Fig. 4.7):

- **Primera capa:** el número k de neuronas coincide con el doble del número de características empleadas para generar la matriz $\mathbf{B} \in \mathbb{R}^{m \times d}$; $k = 2m$ de bases de Fourier. El vector de características de Fourier $\gamma(\mathbf{v})$ se obtiene mapeando las coordenadas tridimensionales de la imagen $(\mathbf{x}_i, \lambda_t)$ siguiendo la Ec. 3.10.

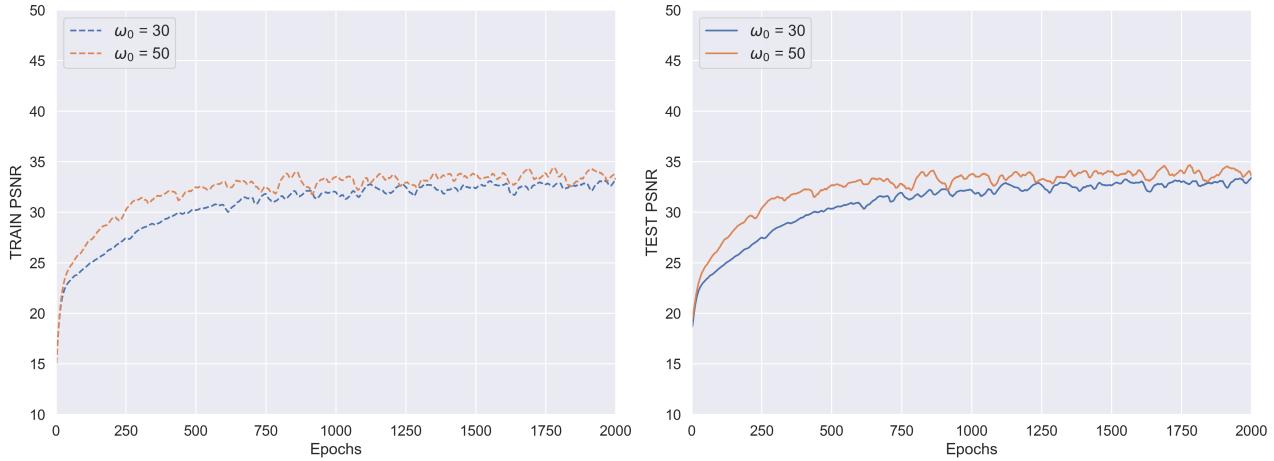


Figura 4.6: Evolución de la PSNR para los datos de entrenamiento (*train*) y prueba (*test*) con diferentes valores de ω a lo largo de las 2000 épocas (*epochs*).

- **Capas ocultas:** el modelo incluye 3 capas ocultas, cada una con 256 neuronas. Utilizamos la función de activación ReLU (Fig. 2.2d) en estas capas, que es especialmente útil para aprender relaciones no lineales en los datos.
- **Capa de salida:** la capa de salida consta de una sola neurona con una función de activación sigmoidal (Fig. 2.2b) que produce la salida final del modelo.

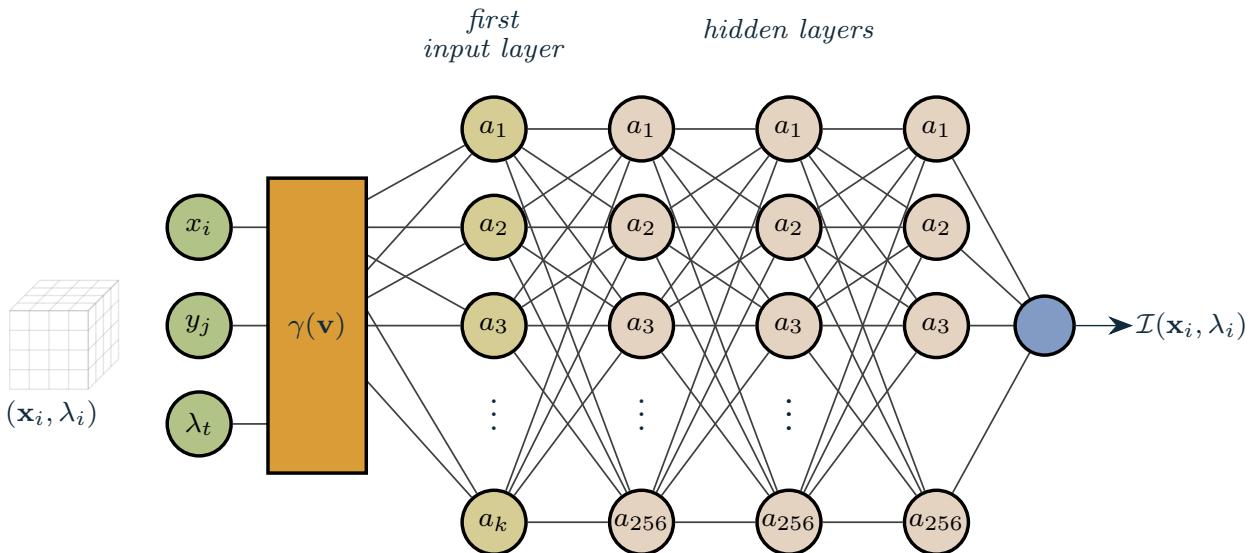


Figura 4.7: Configuración de la red neuronal. La entrada está formada por las coordenadas espaciales y de longitud de onda de cada imagen $(\mathbf{x}_i, \lambda_i)$ normalizadas a la rejilla unidad tridimensional. Posteriormente se mapean estas coordenadas para obtener el vector de características de Fourier (Ec. 3.10) y la primera capa con k neuronas. A continuación se distribuyen el resto de capas ocultas de 256 neuronas cada una y, por último, la capa de salida que contiene el valor de la intensidad del píxel aprendido por la red.

Búsqueda de los hiperparámetros de la red

Queremos determinar si la elección del número de características k tiene un impacto directo en la capacidad del modelo para capturar detalles finos y patrones complejos dentro de las imágenes

sin incurrir en el sobreajuste, por lo que exploraremos tres valores diferentes de k : 128, 256 y 512. La selección de estos valores no es arbitraria: el valor $k = 256$ ya demostró dar buenos resultados (ver apartado 3.3.2) y los valores adicionales $k = 128$ y $k = 512$ nos permitirán evaluar cómo la reducción o el incremento en la complejidad afectan tanto a la calidad de la reconstrucción como al tiempo de entrenamiento. Otro hiperparámetro importante es la desviación estándar de la distribución $N(0, \sigma)$ que se utiliza para seleccionar las entradas de la matriz \mathbf{B} . Como se discutió en el apartado 3.4, la elección adecuada de este parámetro evita el sobreajuste y maximiza la capacidad de generalización del modelo. Para nuestro experimento usaremos el valor con el que obtuvimos el mejor resultado $\sigma = 10$ (ver apartado 3.3.2), y dos valores adicionales no explorados, $\sigma = 0.8$ y $\sigma = 25$. La selección de estos valores tiene como objetivo examinar cómo diferentes grados de transformación en la matriz de mapeo pueden influir en la eficacia del modelo en representar la estructura tridimensional de los datos. Para cada valor de k exploramos todos los valores de σ en un periodo de entrenamiento de 20 épocas. La Tabla 4.2 proporciona esta primera aproximación. Utilizamos la misma métrica (PSNR) que en los experimentos del capítulo 3 para cuantificar la calidad de la reconstrucción, donde un valor más alto indica una mayor fidelidad.

		Mapeo gaussiano		
		$k = 128$	$k = 256$	$k = 512$
σ	PSNR Test	PSNR Test	PSNR Test	PSNR Test
0.8	19.82	19.74	19.87	
10	19.51	19.30	19.74	
25	19.65	19.66	19.67	

Tabla 4.2: Resultados de la relación señal-ruido (PSNR) en el ajuste de los parámetros k y σ para los conjuntos de datos de prueba utilizando los valores de $k = 128$, $k = 256$, $k = 512$ y $\sigma = 0.8$, $\sigma = 10$, y $\sigma = 25$ durante 20 épocas

Observamos que las configuraciones con un $\sigma = 0.8$ alcanzan, aunque no significativamente, los valores más altos de la PSNR en todas las dimensiones de k probadas. Este resultado puede implicar que un mapeo gaussiano más concentrado favorece la representación de las características relevantes de la imagen sin introducir ruido innecesario. Por otro lado, la variación entre los diferentes valores de k es incluso menos relevante, indicando que el incremento en la dimensionalidad no produce mejoras significativas en la PSNR dentro del rango de valores probados. Esto podría sugerir un punto de rendimiento decreciente para valores más altos de σ , donde el aumento de la complejidad del modelo no se traduce necesariamente en una mejor capacidad de generalización. Los resultados destacados en negrita indican las combinaciones de parámetros que lograron el mejor rendimiento, proporcionando una guía para la selección de parámetros en futuros entrenamientos. Para $\sigma = 0.8$ los resultados de la prueba son mejores que los experimentos con σ más grandes. Esto representa un cambio con respecto al caso de la regresión bidimensional en el apartado 3.3.2, donde $\sigma = 10$ proporcionaba los resultados óptimos. Este cambio en la eficacia de σ podría estar relacionado con la introducción del eje de longitud de onda. En una representación tridimensional, este eje podría introducir variaciones y complejidades adicionales en los datos que requieran una desviación estándar más baja. La reducción de σ a 0.8 podría ayudar a capturar mejor estas variaciones, permitiendo una mayor flexibilidad en la adaptación a la estructura tridimensional, donde las frecuencias pueden distribuirse de manera más compleja. Una σ más pequeña podría reflejar una necesidad de sintonizar más finamente la transformación de Fourier para capturar estas frecuencias adicionales.

Sin embargo, quizá podamos aproximarnos a la solución sin excluir los resultado anteriores. En el mapeo gaussiano comúnmente utilizado se emplea una distribución isótropa, donde cada entrada en $\mathbf{B} \in \mathbb{R}^{m \times d; k=2^m}$ se muestran de una distribución normal $\mathcal{N}(0, \sigma)$ y σ se elige de manera uniforme en todas las dimensiones [6]. La forma funcional del mapeo viene dada por la Ec. 3.10, donde la distribución isótropa significa que la frecuencia del espectro de la señal es uniforme en todas las direcciones. En nuestro enfoque, además de con el valor de $\sigma = 0.8$ emplearemos un vector $\boldsymbol{\sigma} = [\sigma_x, \sigma_y, \sigma_\lambda]$, que permite que cada dimensión tenga su propia desviación estándar. Esto refleja la naturaleza única de cada dimensión en el cubo de imágenes. Así, construimos la matriz \mathbf{B} como:

$$\mathbf{B} = \begin{bmatrix} \mathcal{N}(0, \sigma_x) & \mathcal{N}(0, \sigma_y) & \mathcal{N}(0, \sigma_\lambda) \end{bmatrix},$$

donde cada columna corresponde a una dimensión diferente, lo que introduce una mayor flexibilidad y adaptación a las propiedades individuales de cada dimensión, permitiendo a la red aprender las características que son más representativas. Como consecuencia de los resultados de la Tabla 4.2 y el razonamiento anterior, usaremos dos valores de desviación estándar para entrenar la red: el valor escalar $\sigma = 0.8$ y el vector $\boldsymbol{\sigma} = [10, 10, 0.8]$ donde mantenemos el valor de 10 utilizado en la regresión bidimensional para las coordenadas σ_x , σ_y e introducimos el valor de $\sigma_\lambda = 0.8$. Además, para cada valor de la desviación estándar usaremos los valores de $k = 128$ y $k = 512$.

Como vimos en el capítulo 2, la tasa de aprendizaje (*learning rate*) (Ec. 2.15, 2.16) es uno de los hiperparámetros más críticos en el entrenamiento de una red neuronal. Controla el tamaño de los pasos que el modelo toma durante la optimización y, por lo tanto, tiene una fuerte influencia en la convergencia del entrenamiento. Para encontrar su valor óptimo, hemos implementado una búsqueda selectiva dentro de un rango de valores empleando una estrategia de dos etapas:

1. **Búsqueda exponencial (búsqueda gruesa):** en la primera etapa, realizamos una búsqueda en una escala logarítmica, probando valores de la tasa de aprendizaje en forma exponencial. Esto permite una exploración rápida y amplia del espacio, lo que ayuda a identificar el rango más adecuado para una búsqueda más detallada. La búsqueda exponencial es especialmente útil para entender la sensibilidad del modelo a diferentes órdenes de magnitud de la tasa de aprendizaje.
2. **Búsqueda uniforme en un intervalo (búsqueda fina):** en la segunda etapa, llevamos a cabo una búsqueda más focalizada dentro del intervalo identificado en la etapa anterior, utilizando una distribución uniforme. Esta búsqueda permite una exploración más precisa, que ayuda a afinar la selección de la tasa de aprendizaje óptima dentro del rango identificado previamente.

Esta estrategia de dos etapas combina la eficiencia de una búsqueda exponencial con la precisión de una búsqueda uniforme, y permite una selección bien balanceada de la tasa de aprendizaje. La Tabla 4.3 resume los parámetros usados en el experimento.

Los resultados de los experimentos realizados con las características de Fourier, ilustrados en la Tabla 4.4, revelan la precisión de la reconstrucción del conjunto de imágenes. Se observa que el uso de un vector para $\boldsymbol{\sigma}$ que combina diferentes escalas, específicamente [10, 10, 0.8], logra un rendimiento destacado en las métricas de PSNR tanto para los datos de entrenamiento como

Parámetro	Valor
Normalización de las coordenadas	$[0, 1]$
Normalización de los píxeles	$[0, 1]$
Tamaño del recorte de imagen	512×512 píxeles
Transformación de las coordenadas	Mapeo de características de Fourier
Mapeos de características $\gamma(\mathbf{v})$	Fourier gaussiano
Número de características de Fourier (k)	128, 512
\mathbf{B}	$\in \mathbb{R}^{m \times 3; k=2m}$
Desviación estándar en Fourier gaussiano	$\sigma = 0.8, \boldsymbol{\sigma} = [10, 10, 0.8]$
Neuronas en la primera capa	k
Número de capas ocultas	3
Neuronas en las capas ocultas	256
Optimizador	<i>Adam</i>
Tasa de aprendizaje (α)	0.002
Número de épocas	2000
Métrica	PSNR

Tabla 4.3: Resumen de los parámetros utilizados en el experimento de regresión del cubo de datos tridimensional usando un *neural field* con características de Fourier (*Fourier features*).

para los de prueba. El modelo de características de Fourier también supera la PSNR de 30, considerada como un indicador de alta calidad en la reconstrucción de imágenes, lo que indica el buen ajuste del modelo y su capacidad para capturar la complejidad de los datos. La Fig. 4.8

		Mapeo gaussiano				
		$k = 128$		$k = 512$		
σ		PSNR Train	PSNR Test	PSNR Train	PSNR Test	
		0.8	32.77	32.95	33.39	33.42
		[10, 10, 0.8]	34.57	34.52	36.74	36.38

Tabla 4.4: Resultados de la relación señal-ruido (PSNR) en la tarea de regresión de la totalidad de las imágenes para los conjuntos de datos de entrenamiento utilizando, por un lado, los valores de $k = 128$, $k = 512$ y por otro un escalar $\sigma = 0.8$ y un vector $\boldsymbol{\sigma} = [10, 10, 0.8]$.

muestra la evolución de la PSNR a lo largo del entrenamiento para diversas configuraciones de k y σ . La consistencia en la mejora y el mantenimiento de altos valores de la PSNR a través de las épocas enfatiza la robustez del modelo. La configuración con $\sigma = [10, 10, 0.8]$ y $k = 512$ demuestra ser particularmente efectiva, sugiriendo que la introducción de una variedad en la escala del mapeo gaussiano sí puede beneficiar la generalización del modelo, lo que se intuía tímidamente en los valores de la Tabla 4.2.

Por último, la fidelidad visual de las imágenes reconstruidas, que se puede observar en la Fig. 4.9, es objetivamente alta y refleja con precisión las imágenes originales. Esta correspondencia visual avala la capacidad de generalización del modelo, imprescindible para nuestro propósito. Los resultados obtenidos son prometedores y refuerzan la metodología utilizada para optimizar los parámetros del modelo.

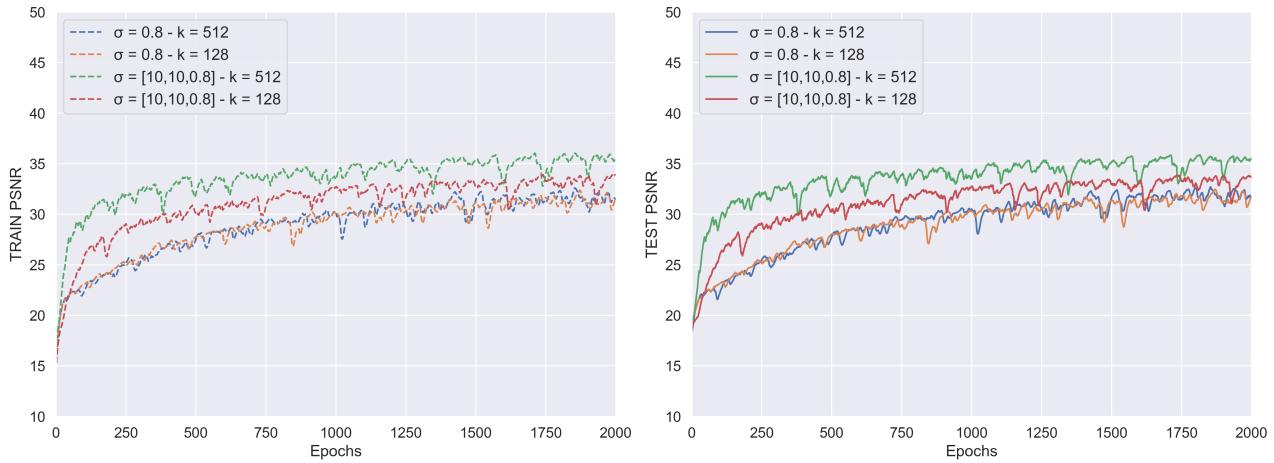


Figura 4.8: Evolución de la PSNR para los datos de entrenamiento (*train*) y prueba (*test*) con diferentes valores de σ y k a lo largo de las 2000 épocas (*epochs*).

4.3. Hacia la Superresolución Espectral

La similitud observada en los resultados, en términos de PSNR, como se muestra en la Fig. 4.10, sugiere una evolución comparable y una notable fidelidad en la reconstrucción de imágenes del conjunto de datos usando ambas técnicas. Esto establece una base sólida para avanzar hacia el siguiente paso: evaluar la capacidad de los modelos para la superresolución espectral. Para ello, realizamos una interpolación de las coordenadas tridimensionales entre cada par de imágenes consecutivas que luego son procesadas por el modelo entrenado. Como resultado, obtenemos una serie de imágenes intermedias; cada una representa un paso específico en la transición espectral entre las imágenes originales. Esta técnica se aplicará a los cuatro modelos presentados en la Fig. 4.10 para analizar cómo interpreta cada uno estas imágenes intermedias.

La Fig. 4.11 muestra la evolución de la intensidad media de píxeles por imagen, comparando los resultados generados por los modelos de SIREN y de *Fourier features* con los valores de las imágenes reales (GT). En la gráfica, los puntos rojos representan la intensidad media de píxeles de las imágenes reales, normalizada a los valores máximos y mínimos de cada imagen. Estos puntos se encuentran en intervalos específicos, correspondientes a cada una de las 21 longitudes de ondas de las imágenes del conjunto de datos. Entre cada par de imágenes consecutivas se han generado ocho imágenes intermedias por cada modelo, lo que resulta en una secuencia detallada que permite observar la transición espectral continua. La presencia de puntos rojos en la gráfica permite hacer una lectura de la precisión de los modelos en la tarea de reconstruir las intensidades de las imágenes. Se observa que el modelo de *Fourier features* con $k = 512$ y $\sigma = [10, 10, 0.8]$ sigue de cerca la tendencia marcada por las imágenes reales, lo que indica una buena capacidad de reconstrucción sin signos evidentes de sobreajuste. Por otro lado, el modelo de SIREN con $\omega_0 = 50$ muestra una concordancia notable con los puntos de las imágenes originales, aunque con fluctuaciones intermedias que podrían interpretarse como variabilidad en la reconstrucción. El modelo de SIREN con $\omega_0 = 30$ y el de *Fourier features* con $k = 128$ también replican las tendencias generales de las imágenes originales, pero con una menor precisión en la reconstrucción. Este comportamiento podría sugerir una menor sensibilidad a las particularidades de cada imagen individual. las fluctuaciones que se observan en la Fig. 4.11 para el caso de SIREN con $\omega_0 = 50$ podrían ser un indicio temprano de un ajuste excesivo

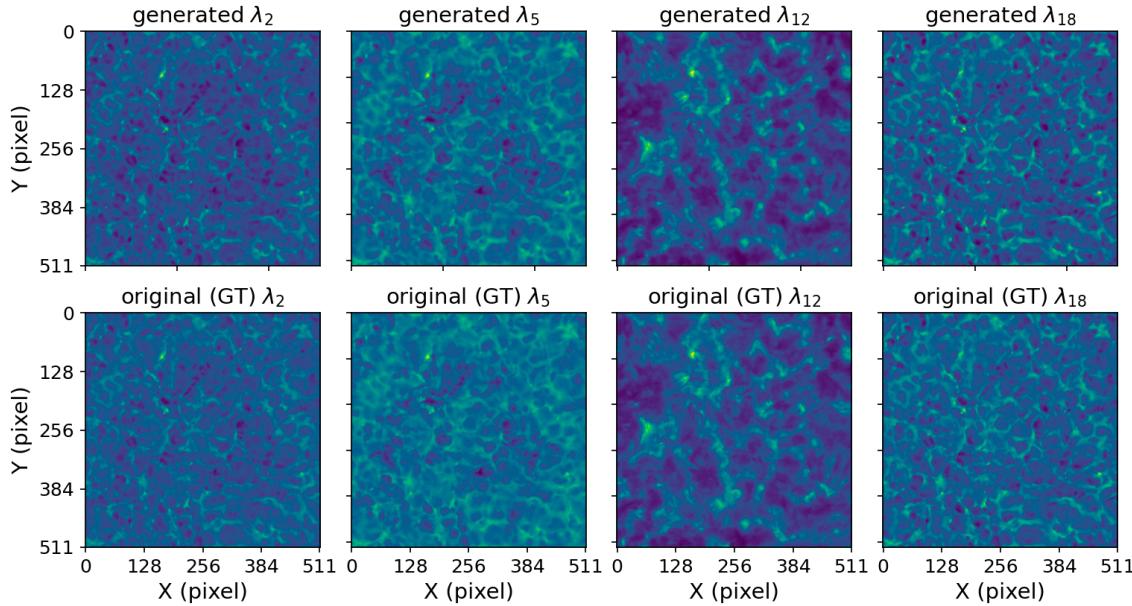


Figura 4.9: Reconstrucción de una porción de 512×512 píxeles de cuatro imágenes del conjunto de datos utilizando la arquitectura de *Fourier features*. Las imágenes superiores representan las reconstrucciones utilizando los valores de $\sigma = [10, 10, 0.8]$, $k = 512$ y las inferiores las originales (Ground truth, GT).

a las especificidades de los datos de entrenamiento, y quizá evidencien cierto sobreajuste del modelo. Aunque la comparativa sugiere que todos los modelos logran seguir la tendencia de las imágenes originales, la elección del modelo óptimo exigirá un equilibrio entre precisión y generalización, con un ligero indicio de que el modelo de *Fourier features* con $k = 512$ podría ofrecer el mejor entre ambos. La Fig. 4.11 revela un comportamiento desigual en la evolución de la intensidad media de los píxeles de los modelos en el intervalo entre las longitudes de onda λ_9 y λ_{11} . Este intervalo muestra una intersección en las trayectorias de los modelos, lo que sugiere una zona de transiciónpectral donde los modelos podrían diferir significativamente en su interpretación de las características de los datos generados. Elegimos este rango para poder realizar un análisis visual de cómo cada modelo gestiona las transiciones espectrales y de validar si las reconstrucciones mantienen coherencia con las imágenes originales en esta región.

La Fig. 4.12 expone una serie de imágenes generadas que demuestran la capacidad de los modelos de SIREN y de *Fourier features* para realizar interpolaciones espectrales entre las longitudes de onda λ_9 y λ_{11} . Esta selección específica ofrece una oportunidad para evaluar la habilidad de los modelos de generar transiciones complejas en los detalles espectrales de las imágenes. Las reconstrucciones obtenidas mediante *Fourier features*, tanto con $k = 512$ (4.12a) como con $k = 128$ (4.12d), exhiben una gran coherencia visual con las imágenes reales (GT) (4.12c), lo que indica una interpolación precisa y una alta fidelidad en la reconstrucción. Pese a que las imágenes correspondientes a *Fourier features* con $k = 128$ (4.12d) muestran una ligera disminución en esta coherencia, se puede sugerir que una menor complejidad del modelo puede ser suficiente para la tarea de superresolución espectral en ciertos rangos de longitud de onda.

Por otro lado, las imágenes generadas por el modelo de SIREN con $\omega_0 = 30$ (4.12b) y $\omega_0 = 50$ (4.12e) reflejan una alta fidelidad en la reconstrucción de las imágenes del conjunto de datos, aunque las intermedias parecen no seguir una evolución visual tan consistente como las generadas por los modelos *Fourier features*. Este fenómeno podría interpretarse como una

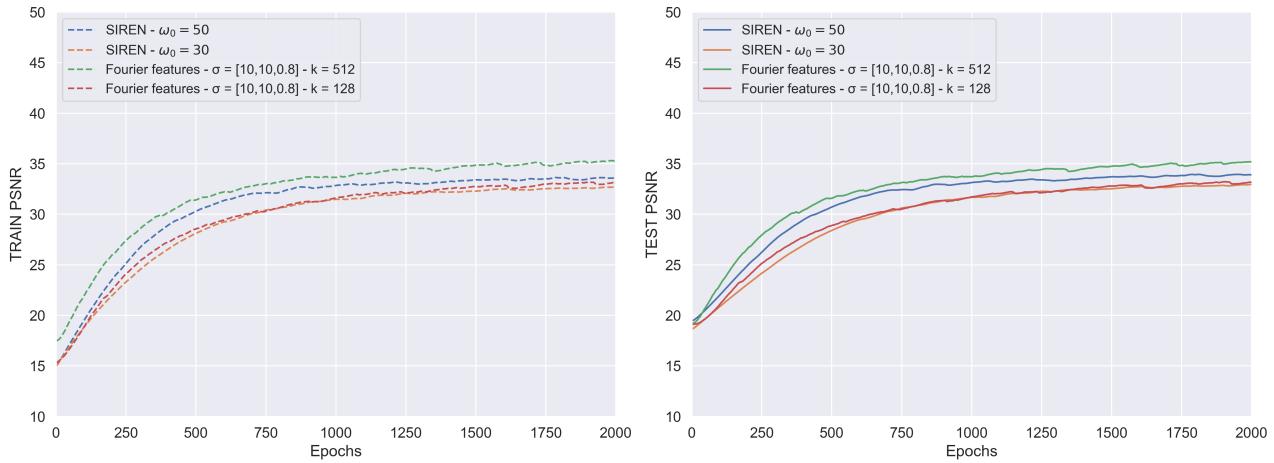


Figura 4.10: Evolución de la PSNR para los datos de entrenamiento (*train*) y prueba (*test*) de los modelos de SIREN con $\omega_0 = 30$ y $\omega_0 = 50$ y de *Fourier features* con $\sigma = [10, 10, 0.8]$ y los valores de $k = 512$ y $k = 128$ a lo largo de las 2000 épocas (*epochs*).

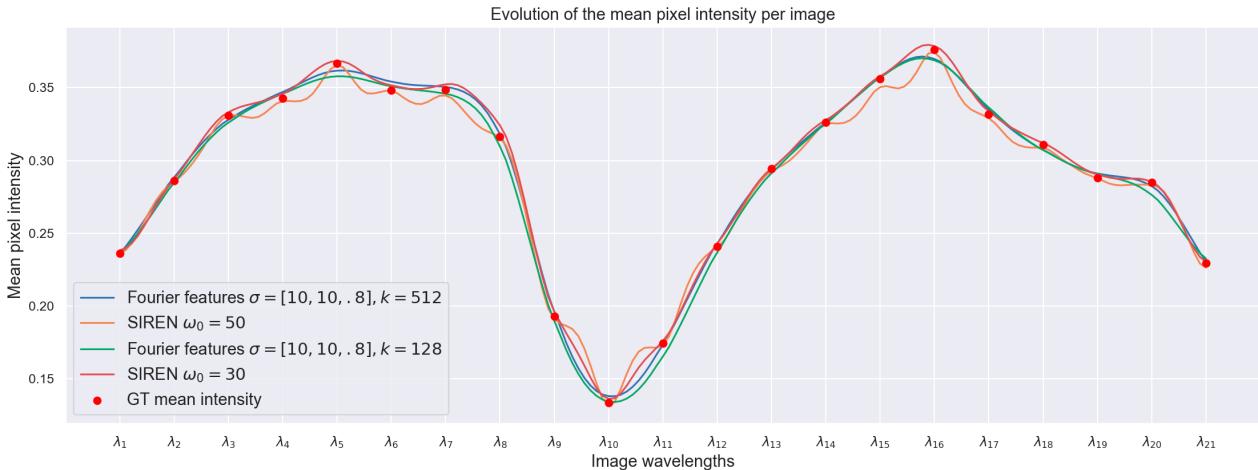


Figura 4.11: Comparativa de la evolución de la intensidad media de píxeles entre las imágenes generadas por los modelos de SIREN y de *Fourier features*, junto con los puntos correspondientes a las medias de intensidad de cada una de las imágenes reales (GT).

tendencia al sobreajuste por el que el modelo se ha ajustado excesivamente a las peculiaridades del conjunto de entrenamiento, comprometiendo la generalización.

Las observaciones cualitativas de las imágenes generadas concuerdan con las métricas cuantitativas previamente analizadas, lo que subraya la importancia de encontrar un equilibrio entre la capacidad del modelo y su habilidad para generalizar, con el objetivo de alcanzar una superresolución espectral con cierto grado de precisión. Las imágenes seleccionadas para la Fig. 4.12 nos han servido para determinar si los patrones y características esenciales de las imágenes originales se conservan en las reconstrucciones y si mantienen cierta coherencia en las representaciones intermedias, lo que indica una buena generalización y adaptabilidad de los modelos a nuevas longitudes de onda dentro del espectro de interés. Al presentar las reconstrucciones generadas para este rango específico, buscábamos verificar visualmente si los modelos están sobreajustando las características de las imágenes de entrenamiento o si en cambio, están extrapolando de manera efectiva, que es el criterio más decisivo para la aplicación práctica de la superresolución espectral. La consistencia visual en este intervalo nos ha mostrado una evidencia adicional para

juzgar la precisión y el realismo de las reconstrucciones generadas por los modelos de SIREN y de *Fourier features*.

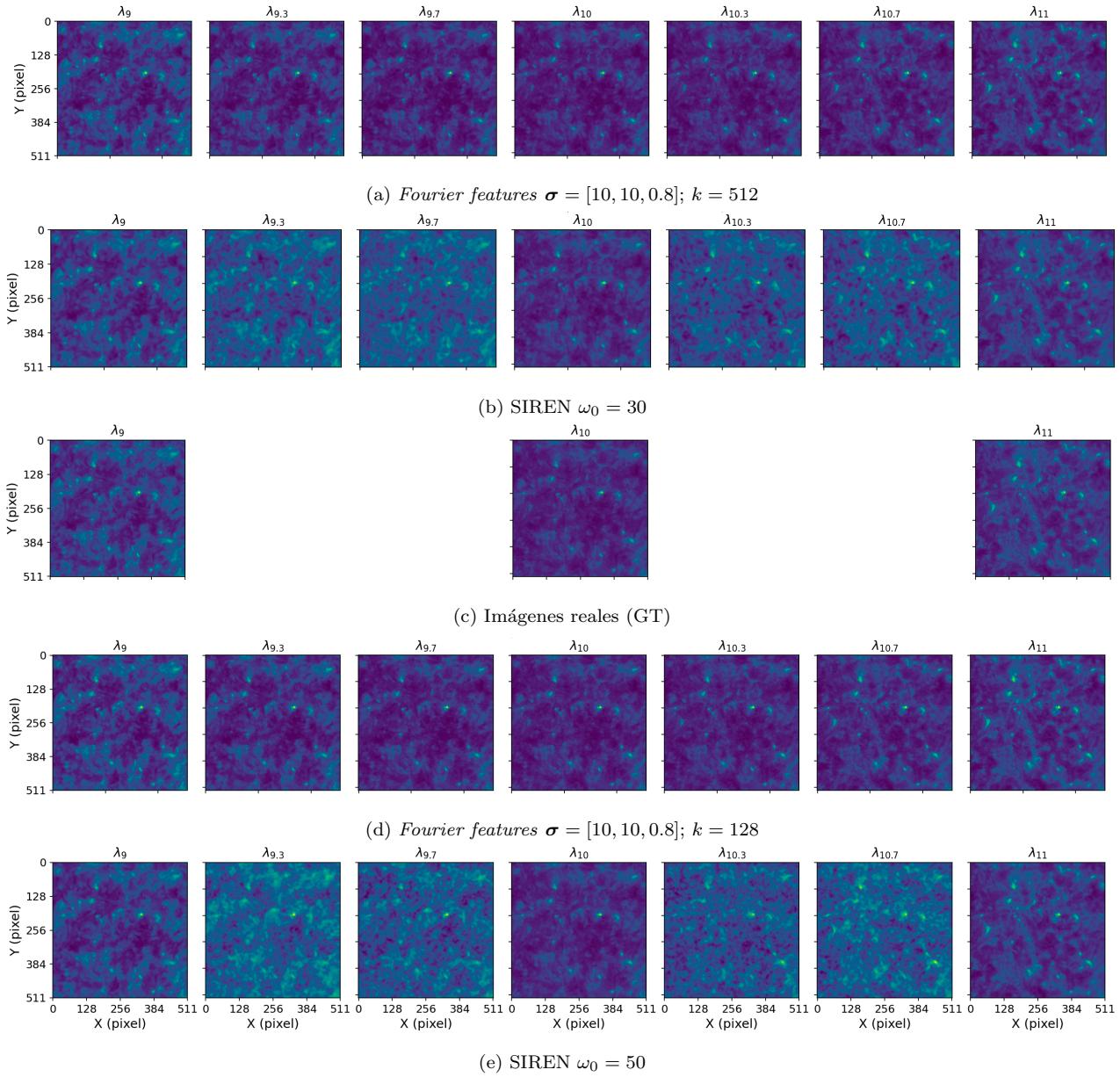


Figura 4.12: Imágenes generadas correspondientes a las longitudes de onda intermedias entre λ_9 y λ_{11} . La selección de este rango se basa en la intersección de las curvas de intensidad de los modelos en la Fig. 4.11, lo que proporciona un caso de prueba para examinar la capacidad de interpolación espectral y la fidelidad de la reconstrucción.

4.4. Conclusiones

El presente Proyecto de Fin de Grado ha estado motivado por la necesidad de obtener imágenes precisas a distintas longitudes de onda. Capturar estos datos es fundamental en diversos campos científicos, desde la astrofísica hasta la biomedicina, pero las metodologías convencionales nos limitan a longitudes de onda discretas, dejando vacíos significativos en nuestra capacidad para el

análisis y la compresión de numerosos fenómenos. Este proyecto ha intentado llenar esas lagunas para proporcionar una imagen más completa y detallada, específicamente de la fotosfera solar.

En este proyecto, trabajamos con un conjunto de datos espectrales preexistentes, concretamente la línea de absorción de Fe I $\lambda 6302$, obtenidos con el instrumento CRISP en el Swedish Solar Telescope en La Palma. Estos datos consisten en una serie de 21 imágenes ($\lambda_1 - \lambda_{21}$) pseudomonocromáticas de 966×964 píxeles (Fig. 4.1). El enfoque central del proyecto fue integrar estos datos en los pesos de una red neuronal. Para ello, se emplearon arquitecturas de redes neuronales totalmente conectadas y se utilizaron algoritmos de descenso por gradiente en el entrenamiento. El objetivo principal ha sido evaluar la posibilidad de realizar interpolaciones entre imágenes y alcanzar la superresoluciónpectral utilizando campos neuronales, conocidos como representaciones implícitas.

El primer paso en nuestro proyecto consistió en la replicación y evaluación de los experimentos basados en las técnicas de SIREN [5] y características de Fourier (*Fourier features*) [6], aplicadas a nuestras imágenes solares. El objetivo es explorar y comprender el comportamiento de estos métodos avanzados de aprendizaje profundo utilizando nuestros datos específicos. Para ello, seleccionamos una imagen (λ_{12}) de nuestro conjunto de 21 imágenes y extrajimos una región central de 512×512 píxeles. Esta decisión se tomó para mantener un equilibrio entre la complejidad computacional y la habilidad para capturar los detalles de la imagen, siguiendo el método usado por otros autores [6, 5] en cuanto al tamaño de la muestra.

Tras seleccionar esta región, procedimos a aplicar los modelos de SIREN y características de Fourier, con el objetivo de investigar la eficiencia de estas técnicas en la reconstrucción y regresión de imágenes bidimensionales y validarlas en nuestro conjunto específico de datos. Esta aproximación nos permitió profundizar en las diferencias en el rendimiento, la sensibilidad a los parámetros e hiperparámetros, y la capacidad de generalizar a partir de los datos de entrenamiento. Los resultados obtenidos en los experimentos indican la viabilidad de estas técnicas en el contexto de aplicaciones de superresoluciónpectral.

En la metodología empleada con SIREN, el hiperparámetro ω_0 juega un papel clave, ya que determina la frecuencia de la función de activación sinusoidal utilizada en la red. Un valor más elevado de ω_0 implica una mayor capacidad para captar variaciones de alta frecuencia en los datos, aspecto esencial para reproducir con precisión los detalles finos de las imágenes. Observamos que al incrementar ω_0 , la relación señal-ruido en el conjunto de entrenamiento experimentaba un aumento progresivo, alcanzando su punto máximo de PSNR = 49.13 para $\omega_0 = 100$. No obstante, en el conjunto de prueba, el valor más adecuado se situó en $\omega_0 = 50$, con una PSNR = 38.50 (Tabla 3.2). Este resultado refleja un equilibrio entre la capacidad del modelo para aprender los detalles de los datos de entrenamiento y su habilidad para generalizar correctamente a nuevos datos no vistos anteriormente.

En el ámbito de las características de Fourier, el hiperparámetro de mayor relevancia es σ , que actúa como la desviación estándar de la distribución gaussiana utilizada para muestrear la matriz \mathbf{B} en el mapeo de características. Este parámetro es crucial, ya que regula la amplitud de la variabilidad en la transformación de las coordenadas de entrada, afectando de manera determinante la forma en que el modelo interpreta y procesa las frecuencias de los datos. Un valor adecuado de σ , como hemos comprobado con $\sigma = 10$, facilita que el modelo capte eficazmente los detalles de la imagen, reflejado en una PSNR = 39.02 en el conjunto de prueba. Por otro lado, un σ excesivamente alto, como el valor $\sigma = 100$, puede distorsionar significati-

vamente esta interpretación, conduciendo a un sobreajuste en los datos de entrenamiento y a una reducción de la capacidad del modelo para generalizar a datos nuevos, como se evidencia en una PSNR = 17.85 en el conjunto de prueba (Tabla 3.4).

En definitiva, los experimentos llevados a cabo con las técnicas de SIREN y características de Fourier proporcionaron resultados que refuerzan su potencial para aplicaciones de superresolución espectral, particularmente visibles en las métricas de PSNR mostradas en la Fig. 3.7. La capacidad de ambos modelos para generalizar a partir de los datos de entrenamiento nos permitió avanzar hacia nuestro objetivo de sintetizar imágenes intermedias que no se encuentran en el cubo de imágenes observadas y comprobar la versatilidad de los modelos para interpretar y complementar el espectro de datos disponibles. Por último, las evaluaciones cuantitativas, reflejadas en valores de PSNR, fueron coherentes con la calidad visual de las imágenes reconstruidas (Fig. 3.8). Las imágenes generadas demostraron una alta fidelidad reproduciendo los detalles complejos de la fotosfera solar.

Con los resultados obtenidos en la regresión de imágenes bidimensionales, abordamos el siguiente paso de nuestro proyecto: intentar compactar el cubo de datos tridimensional para llenar las lagunas espetrales producidas por el muestreo discreto (Fig. 4.2). Uno de los retos fue extender el experimento manteniendo inalterada la arquitectura de la red, preservando así la integridad de las representaciones implícitas. Por tanto, el trabajo en esta fase se centró en la preparación de los datos para su adecuada alimentación a la red y en el ajuste de los hiperparámetros. Los modelos debían reconstruir las imágenes con precisión y proporcionar transiciones suaves entre las imágenes generadas a lo largo del eje de la longitud de onda.

El preprocessado de las imágenes consistió en normalizar las intensidades al intervalo [0, 1] para ajustar los valores en la misma escala, facilitando así la convergencia durante el entrenamiento y mejorando la estabilidad y rendimiento del modelo. Adicionalmente, creamos una malla tridimensional que mapeaba todas las coordenadas (x_i, y_j, λ_t) a valores dentro del intervalo [0, 1], lo que permitió a la red aprender patrones tanto espaciales como espetrales. Esta preparación de los datos, junto con la selección de hiperparámetros, constituyeron nuestro esfuerzo inicial por lograr que las redes neuronales reprodujeran con precisión el conjunto de las 21 imágenes pseudomonocromáticas.

Los resultados de la PSNR obtenidos con SIREN superaron el umbral de 30 tanto para $\omega_0 = 30$ como para $\omega_0 = 50$, como se puede observar en la Tabla 4.1, donde la PSNR alcanzó 35.02 para $\omega_0 = 50$ con el conjunto de prueba, indicando una excelente reproducción de las imágenes originales. La validez de la red SIREN quedó también demostrada en las comparaciones visuales de las imágenes reconstruidas con respecto a las originales, como muestra la Fig. 4.5. La capacidad del modelo para capturar con fidelidad la información espectral y espacial queda patente en la alta similitud visual entre las imágenes generadas y las reales. Además, la Fig. 4.6 muestra la evolución de la PSNR a lo largo del entrenamiento, que demuestra una estabilidad y consistencia en el rendimiento del modelo, lo que indica su capacidad de generalización, un aspecto importante para la interpolación espectral continua en el contexto de la superresolución.

En nuestro experimento con Fourier Features, la optimización de los hiperparámetros fue esencial para adaptar el modelo a la complejidad tridimensional de los datos. Nos centramos en determinar cómo el número de características k y la desviación estándar σ afectaban a la capacidad del modelo para capturar detalles y patrones complejos en las imágenes. Seleccionamos valores de k (128, 256 y 512) para evaluar su impacto en la calidad de la reconstrucción y en el

tiempo de entrenamiento. Además, probamos varios valores de σ (0.8, 10 y 25) para examinar cómo la transformación de Fourier influía en la representación de los datos tridimensionales. Un hallazgo importante fue que un $\sigma = 0.8$ proporcionaba mejores resultados que valores más altos, indicando que un mapeo gaussiano más concentrado es más adecuado para la nueva estructura tridimensional. Este descubrimiento nos llevó a considerar σ no como un valor único, sino como un vector $\boldsymbol{\sigma} = [\sigma_x, \sigma_y, \sigma_\lambda]$, lo que permite a cada dimensión tener su propia desviación estándar. Esta aproximación refleja la naturaleza única de cada dimensión del cubo de imágenes y ofrece una mayor flexibilidad para capturar las características representativas de cada eje. Para determinar la tasa de aprendizaje utilizamos una estrategia que incluyó una búsqueda exponencial para identificar un rango adecuado y una búsqueda uniforme para una selección precisa dentro de ese rango. Todos los parámetros seleccionados se detallan en la Tabla 4.3. Con ellos procedimos al entrenamiento de la red.

El uso de un vector para $\boldsymbol{\sigma}$ que combinaba diferentes escalas, en particular $\boldsymbol{\sigma} = [10, 10, 0.8]$, y el número de características $k = 512$, obtuvo el mejor rendimiento, que se reflejó en las métricas de PSNR obtenidas tanto para los datos de entrenamiento como para los de prueba. Los valores de PSNR superaron el umbral de 30 llegando a obtener un $\text{PSNR} = 36.38$, lo que confirmó la capacidad del modelo para capturar la complejidad de los datos de manera efectiva. Además, la fidelidad visual de las imágenes reconstruidas, mostrada en la Fig. 4.9, fue objetivamente alta. Esta correspondencia visual no solo respalda la calidad de la reconstrucción, sino que también avala la capacidad de generalización del modelo.

Llegamos por fin al momento de realizar las interpolaciones para evaluar cómo cada modelo maneja la generación de imágenes intermedias en el eje de la longitud de onda. Tras obtener resultados comparables en términos de PSNR con ambas técnicas, como se ilustra en la Fig. 4.10, nos dispusimos a examinar su capacidad para la superresolución espectral. Implementamos una técnica de interpolación de coordenadas tridimensionales entre pares de imágenes consecutivas, que fueron procesadas luego por los modelos entrenados. Con esto generamos las imágenes intermedias, representando pasos específicos en la transición espectral entre las imágenes originales. Esta técnica se aplicó a los cuatro modelos discutidos anteriormente, proporcionando una perspectiva detallada de cómo cada uno interpreta estas imágenes intermedias. La Fig. 4.11 compara la evolución de la intensidad media de los píxeles entre los modelos y las imágenes reales (GT), y revela la precisión de los modelos para reconstruir las intensidades de las imágenes. Observamos que el modelo de Fourier Features con $k = 512$ y $\boldsymbol{\sigma} = [10, 10, 0.8]$ sigue de cerca la tendencia de las imágenes reales, mostrando una alta capacidad de reconstrucción sin signos de sobreajuste. El modelo SIREN con $\omega_0 = 50$ también muestra una buena correspondencia con los valores de las imágenes originales, aunque con algunas fluctuaciones que podrían interpretarse como una variabilidad en la reconstrucción. Los modelos con parámetros menos complejos, como SIREN con $\omega_0 = 30$ y Fourier Features con $k = 128$, replican las tendencias generales, pero con menor precisión.

La generación de imágenes intermedias en el intervalo entre λ_9 y λ_{11} nos permitió evaluar visualmente la capacidad de los modelos para manejar transiciones espirituales complejas. Las imágenes reconstruidas por Fourier Features con $k = 512$ (4.12a) y $k = 128$ (4.12d), así como por SIREN con $\omega_0 = 30$ (4.12b) y $\omega_0 = 50$ (4.12e), se compararon con las imágenes reales (4.12c). Esta comparación reveló que Fourier Features con $k = 512$ logra una interpolación más suave y una alta fidelidad en la reconstrucción, mientras que el modelo con $k = 128$ muestra una ligera disminución en coherencia. Por su parte, las imágenes generadas por SIREN reflejan una alta fidelidad, aunque con menos consistencia visual en las transiciones espirituales que

Fourier Features, sugiriendo una posible tendencia al sobreajuste en ciertos casos.

Reflexionando sobre el camino recorrido, este Proyecto de Fin de Grado nos ha dado la oportunidad de adentrarnos en los campos neuronales. Hemos experimentado con técnicas avanzadas, como SIREN y Fourier features, que realzan el rendimiento de los algoritmos de descenso por gradiente aplicándolas a una tarea compleja como la interpolación de imágenes espectrales. Hemos intentado abordar los desafíos que presenta esta tarea y hemos podido evaluar la capacidad de los modelos para generar reconstrucciones de alta fidelidad y representaciones intermedias coherentes. Hemos confirmado que la selección cuidadosa de los parámetros, como la desviación estándar en el mapeo de Fourier y la frecuencia en la arquitectura SIREN, es fundamental para capturar la variabilidad espectral y espacial de los datos. La adaptabilidad de estos parámetros es crucial, especialmente al extender nuestras consideraciones a la dimensión adicional de la longitud de onda. Las visualizaciones generadas han proporcionado una herramienta esencial para evaluar cualitativamente la coherencia de la superresolución espectral. Al seleccionar un intervalo crítico basado en la intersección de las curvas de las intensidades medias, hemos podido examinar la habilidad de los modelos para manejar transiciones espectrales complejas y mantener la coherencia de las características espirituales fundamentales.

La realización de este trabajo ha enriquecido nuestra comprensión teórica y práctica de las redes neuronales y de la superresolución espectral y muy probablemente sus resultados mejorarían si profundizáramos en algunas de las cuestiones que nos han asaltado durante su desarrollo: ¿Cuál es la relación teórica, más allá de la mera intuición, entre las técnicas de *Fourier features* y SIREN en el contexto de los campos neuronales? ¿Qué otras técnicas de optimización de los hiperparámetros podríamos aplicar aparte de las aquí empleadas? ¿Cómo podríamos evaluar y formalizar la preparación y estructuración de los datos para optimizar el rendimiento de los modelos?

Finalmente, esperamos que las metodologías ensayadas en nuestros experimentos se apliquen a una amplia gama de conjuntos de datos, con el objetivo de evaluar la capacidad de generalización y la efectividad de las técnicas propuestas. Confiamos en que este Proyecto de Fin de Grado sirva tanto para estudiantes como para futuros desarrollos en el ámbito de la superresolución espectral, poniendo de relieve no solo el potencial considerable de los enfoques actuales, sino también la promesa de nuevos descubrimientos en este campo fascinante y en constante evolución.

Bibliografía

- [1] P. Bootcamp, “Section: Fabry-perot interferometer,” 2019, [Online]. [Online]. Available: <http://www.physicsbootcamp.org/section-fabry-perot-interferometer.html>
- [2] A. Tritschler, W. Schmidt, K. Langhans, and T. Kentischer, “High-resolution solar spectroscopy with tesos - upgrade from a double to a triple system,” *Solar Physics*, vol. 211, p. 17, 2002.
- [3] T. Kentischer, W. Schmidt, M. Sigwarth, and M. von Uexkuell, “Teson, a double fabry-perot instrument for solar spectroscopy,” *Astronomy & Astrophysics*, vol. 340, p. 569, 1998.
- [4] Y. Xie, T. Takikawa, S. Saito, O. Litany, S. Yan, N. Khan, F. Tombari, J. Tompkin, V. Sitzmann, and S. Sridhar, “Neural fields in visual computing and beyond,” *arXiv preprint arXiv:2111.11426v4*, 2022. [Online]. Available: <https://arxiv.org/abs/2111.11426v4>
- [5] V. Sitzmann, J. N. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein, “Implicit neural representations with periodic activation functions,” in *Proc. NeurIPS*, 2020.
- [6] M. Tancik, P. P. Srinivasan, B. Mildenhall, S. Fridovich-Keil, N. Raghavan, U. Singhal, R. Ramamoorthi, J. T. Barron, and R. Ng, “Fourier features let networks learn high frequency functions in low dimensional domains,” in *Advances in Neural Information Processing Systems (NeurIPS)*. Curran Associates, Inc., 2020. [Online]. Available: <http://arxiv.org/abs/2006.10739v1>
- [7] Q. Huynh-Thu and M. Ghanbari, “Scope of validity of psnr in image/video quality assessment,” *Electronics letters*, vol. 44, no. 13, pp. 800–801, 2008.
- [8] Swedish Solar Telescope Team, “Crisp - crisp imaging spectro-polarimeter,” Swedish Solar Telescope, La Palma, 2006. [Online]. Available: <https://dubshen.astro.su.se/wiki/index.php/CRISP>
- [9] M. Faraday, “On the physical character of the lines of magnetic force,” *Philosophical Transactions of the Royal Society of London*, vol. 142, pp. 1–20, 1852.
- [10] J. C. Maxwell, “A dynamical theory of the electromagnetic field,” *Philosophical Transactions of the Royal Society of London*, vol. 155, pp. 459–512, 1865.
- [11] A. Einstein, “Die feldgleichungen der gravitation,” *Sitzungsberichte der Preussischen Akademie der Wissenschaften, Berlin*, pp. 844–847, 1915.
- [12] P. A. M. Dirac, “The quantum theory of the emission and absorption of radiation,” *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 114, no. 767, pp. 243–265, 1927.
- [13] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

- [14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012, pp. 1097–1105.
- [15] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, “Language models are unsupervised multitask learners,” *Openai Blog*, vol. 1, no. 8, 2019.
- [16] W. S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity,” *The Bulletin of Mathematical Biophysics*, vol. 5, pp. 115–133, 1943.
- [17] G. Cybenko, “Approximation by superpositions of a sigmoidal function,” *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.
- [18] K. Hornik, “Approximation capabilities of multilayer feedforward networks,” *Neural networks*, vol. 4, no. 2, pp. 251–257, 1991.
- [19] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014.
- [20] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.
- [21] C. Dong, C. C. Loy, K. He, and X. Tang, “Image super-resolution using deep convolutional networks,” in *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 2, 2016, pp. 295–307.
- [22] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, “On the importance of initialization and momentum in deep learning,” *International conference on machine learning*, vol. 28, pp. 1139–1147, 2013.
- [23] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.
- [24] J. Duchi, E. Hazan, and Y. Singer, “Adaptive subgradient methods for online learning and stochastic optimization,” *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.
- [25] T. Tieleman and G. Hinton, “Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude,” *COURSERA: Neural Networks for Machine Learning*, vol. 4, no. 2, pp. 26–31, 2012.
- [26] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [27] S. Ruder, “An overview of gradient descent optimization algorithms,” *arXiv preprint arXiv:1609.04747*, 2016.
- [28] L. Bottou, F. E. Curtis, and J. Nocedal, “Optimization methods for large-scale machine learning,” *SIAM Review*, vol. 60, no. 2, pp. 223–311, 2018.
- [29] A. Halevy, P. Norvig, and F. Pereira, “The unreasonable effectiveness of data,” *IEEE Intelligent Systems*, vol. 24, no. 2, pp. 8–12, 2009.

- [30] S. García, J. Luengo, and F. Herrera, *Data Preprocessing in Data Mining*. Springer, 2016.
- [31] Z. Reitermanova, “Data splitting,” *WDS’10 Proceedings of Contributed Papers*, pp. 31–36, 2010.
- [32] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.
- [33] D. M. Hawkins, “The problem of overfitting,” *Journal of chemical information and computer sciences*, vol. 44, no. 1, pp. 1–12, 2004.
- [34] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2009.
- [35] J. Fourier, *The Analytical Theory of Heat*. Cambridge University Press, 1822.
- [36] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1994. [Online]. Available: <https://books.google.com/books?id=JZ3vQgAACAAJ>
- [37] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., 2019.
- [38] NVIDIA, “Nvidia tesla v100 gpu architecture,” <https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/tesla-v100/pdf/437451-Tesla-V100-DS-NV-US-WEB.pdf>, 2020.
- [39] C. Li, H. Zheng, and W. Tompkins, “Data preprocessing for machine learning,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 3, pp. 609–620, 2015.
- [40] V. Nair and G. Hinton, “Rectified linear units improve restricted boltzmann machines,” in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 807–814.
- [41] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.
- [42] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.
- [43] R. Bellman, *Adaptive Control Processes: A Guided Tour*. Princeton University Press, 1961.
- [44] A. Rahimi and B. Recht, “Random features for large-scale kernel machines,” in *Advances in neural information processing systems*, 2007, pp. 1177–1184.
- [45] J. Pons and X. Serra, “End-to-end learning for music audio,” in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 696–700.
- [46] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, 2009.

- [47] R. C. Gonzalez, R. E. Woods, and S. L. Eddins, *Digital image processing*. Pearson Prentice Hall, 2009.
- [48] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein, “Siren: Implications of representational differences between neural and sinusoidal activations,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.
- [49] N. Doshi, “Understanding activation functions in neural networks,” *Medium, Towards Data Science*, 2018. [Online]. Available: <https://towardsdatascience.com/understanding-activation-functions-in-neural-networks-9491262884e0>
- [50] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, 1998.
- [51] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.
- [52] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.
- [53] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge, MA, USA: MIT Press, 2009.
- [54] “Google colaboratory,” Plataforma en línea, Google, 2021, accedido: 2024-04-10. [Online]. Available: <https://colab.research.google.com/>
- [55] M. Riedmiller and H. Braun, “The backpropagation algorithm: A tutorial for practical application,” *Neural networks*, vol. 6, no. 4, pp. 593–603, 1993.

Lista de abreviaturas y acrónimos

Adagrad Algoritmo de optimización Adagrad (Adaptive Gradient). 22

Adam Algoritmo de optimización Adam (Adaptive Moment Estimation). 22, 28, 31, 32, 40, 45

Ca II Calcio ionizado. 3, XVII

CRISP CRisp Imaging SpectroPolarimeter. 6

Fe I Hierro neutro. 5, 6, 37–39, III, IX, XVII, XIX

FF Fourier features (características de Fourier). 34, 35, XVIII

GD Descenso de gradiente. 22

línneas H y K Líneas de absorción en el espectro del Sol asociadas con el calcio ionizado (Ca II). 3, XVII

MAE Error absoluto medio. 15

MSE Error cuadrático medio. 14, 15, 28, 31, 32

PFG Proyecto Fin de Grado. 4, 5, III, IX

PSNR Relación señal-ruido (Peak signal-to-noise ratio). 5, 28–34, 40–46, 48, 50–52, III, IX, XV, XVIII, XIX

ReLU Rectified Linear Unit (función de rectificado lineal). 11, 12

RMSprop Algoritmo de optimización RMSprop (Root Mean Square Propagation). 22

SGD Descenso de gradiente estocástico. 22

SIREN SINusoidal REpresentation Networks (Representaciones implícitas sinusoidales). 5, 19, 23–31, 33, 35, 37, 40, 41, 46–53, III, V, IX, XI, XV, XVIII–XX

tanh Función tangente hiperbólica. 11, 12

TESOS Triple Etalon Solar Spectrograph (Espectrógrafo solar de triple etalón). 5

Apéndice A

Análisis del rendimiento de los modelos

Introducción

En este anexo analizamos el rendimiento de los modelos implementados para los experimentos llevados a cabo. El coste temporal, es decir, el tiempo computacional necesario para entrenar y ejecutar estos modelos, y el coste espacial, relacionado con el almacenamiento de sus parámetros y las estructuras de datos intermedias, son importantes para evaluar su viabilidad y su eficiencia en aplicaciones prácticas.

Normalmente, el análisis del coste de los algoritmos se realiza utilizando la notación Big O (O), que permite establecer una relación conceptual entre el tamaño de la entrada n y la complejidad, tanto en términos de operaciones como de memoria. Sin embargo, aunque la notación Big O ofrece una visión útil sobre el comportamiento asintótico de los algoritmos, en la práctica, se emplean métodos que detallan cómo afectan otros factores —la arquitectura de la red, las funciones de activación, los algoritmos de optimización, los métodos de regularización y el uso de parallelización en hardware específico— a la complejidad del entrenamiento. Por tanto, comenzaremos el análisis describiendo brevemente la infraestructura utilizada, luego mostraremos un cálculo aproximado del coste temporal y del coste espacial requeridos para la ejecución de los modelos y, posteriormente, ofrecemos los datos empíricos obtenidos mediante PyTorch Profiler. Esto proporcionará una visión detallada y práctica del rendimiento real de los modelos en nuestros experimentos.

A.1. Infraestructura utilizada

Para el desarrollo de este Proyecto de Fin de Grado, hemos utilizado Google Colab [54], una plataforma de investigación que facilita la ejecución de código Python en el navegador a través de un entorno de Jupyter notebook y sin necesidad de configuración previa. Esta herramienta está basada en la infraestructura de la nube de Google y proporciona acceso a recursos computacionales avanzados, como CPUs y GPUs. Se usa con frecuencia en el ámbito académico y de investigación para aplicaciones de machine learning y ciencia de datos debido a su accesibilidad y a su sencillez.

Google Colab ofrece varios tipos de recursos de hardware, desde CPUs estándar hasta GPUs como las NVIDIA Tesla T4, L4, V100 y A100. Al basarse en Jupyter Notebooks permite escribir, ejecutar, guardar y compartir código de manera sencilla. Además, se integra con Google Drive y otros servicios y permite la carga y almacenamiento de datos directamente.

Google Colab ofrece un modelo de contratación basado en el uso de *unidades de cómputo* (UCs). Los precios varían según la cantidad de recursos consumidos, como CPU, GPU o RAM

y dependen de la zona geográfica desde la que se contrate el servicio. En las Islas Canarias, España, el coste de 500 unidades de cómputo es de 42,25 euros. En la Tabla A.1 se comparan los costes en unidades de computación con las horas de uso correspondientes y su coste en euros así como las características de cada modelo.

Infraestructura	Coste (uic/hora)	Horas de uso	Precio (euros/hora)	TFLOPS Operaciones/seg.	RAM sistema	RAM GPU
CPU	0.07	7.142,86	0.01	-	14 GB	-
	0.13	3.846,15	0.01	-	55 GB	-
T4 GPU	1.76	284,09	0,15	8.1 TFLOPS	14 GB	15 GB
	1.84	271,74	0,16	-	55 GB	-
L4 GPU	4.82	103,73	0,41	7.4 TFLOPS	64 GB	22.5 GB
V100 GPU	4.82	103,73	0,41	14.1 TFLOPS	14 GB	16 GB
	4.91	101.83	0,41	-	55 GB	-
A100 GPU	11.77	42.48	0,99	19.5 TFLOPS	90 GB	40 GB

Tabla A.1: Comparativa de infraestructuras disponibles en Google Colab, incluyendo los costes operativos y las capacidades de procesamiento.

A.2. Coste Temporal

El **coste temporal** se refiere a la cantidad de tiempo que tarda un algoritmo en completarse. Este se mide generalmente en términos de la cantidad de operaciones fundamentales (comparaciones, asignaciones, accesos a estructuras de datos, etc.) que realiza el algoritmo en función del tamaño de entrada n .

SIREN

Para el modelo SIREN representado en la Fig. 3.1 y en la Fig. 4.4 , este coste se analiza en función del número de operaciones requeridas por cada capa durante la propagación hacia adelante y hacia atrás a través de la red.

Durante la propagación hacia adelante, cada capa densa realiza operaciones matriciales para transformar las entradas en salidas. La complejidad de una operación matricial es proporcional al número de multiplicaciones y sumas realizadas.

- Para la primera capa, con una entrada de tamaño $n = 3$ y $m = 256$ neuronas, el número total de operaciones es $n \times m = 3 \times 256 = 768$.
- Para cada una de las capas intermedias, con $n = m = 256$, el número total de operaciones es $n \times m = 256 \times 256 = 65.536$.
- Para la capa de salida, que convierte una entrada de $n = 256$ a una salida de $m = 1$, el número total de operaciones es $n \times m = 256 \times 1 = 256$.

Sumando el número total de operaciones de todas las capas, obtenemos una estimación de las operaciones totales requeridas por entrada. Es decir, para un único conjunto de coordenadas (x_i, y_i, λ_i) , el modelo realiza aproximadamente $768 + 65536 \times 2 + 256 = 132.096$ operaciones.

Durante la propagación hacia atrás, necesaria para el entrenamiento del modelo, en cada capa se calculan los gradientes con respecto a sus pesos y sus sesgos. Esto implica realizar operaciones adicionales que, en general, son k veces las operaciones realizadas durante la propagación hacia adelante ($k \geq 2$) [55]. Teniendo en cuenta ambas, el número total de operaciones, en el mejor de los casos ($k = 2$), se aproxima a $132.096 \times 3 = 396.288$ para un paso del entrenamiento de una muestra.

Características de Fourier

Para el modelo de Fourier features que presenta la arquitectura secuencial mostrada en la Fig. 3.4 y en la Fig. 4.7 el cálculo se realiza de la misma manera con la salvedad de que la primera capa puede estar compuesta de 128 o 512 neuronas en función del mapeo realizado. Si tenemos en cuenta una primera capa con 512 neuronas:

- Para la primera capa, con una entrada de tamaño $n = 512$ y $m = 256$ neuronas, el número total de operaciones es $n \times m = 512 \times 256 = 131.072$.
- Para cada una de las capas intermedias, con $n = m = 256$, el número total de operaciones es $n \times m = 256 \times 256 = 65.536$.
- Para la capa de salida, que convierte una entrada de $n = 256$ a una salida de $m = 1$, el número total de operaciones es $n \times m = 256 \times 1 = 256$.

El número total de operaciones matriciales para la propagación hacia adelante es aproximadamente de $131072 + 65.536 \times 2 + 256 = 262.400$. Considerando que el número de operaciones de la propagación hacia atrás sea aproximadamente el doble ($k = 2$), el coste temporal total para entrenar una muestra es aproximadamente de $262.400 \times 3 = 787.200$ operaciones.

A.3. Coste Espacial

El **coste espacial** se refiere a la cantidad total de memoria requerida por un algoritmo durante su ejecución. Esto incluye tanto el espacio estático (el código y las variables fijas) como el dinámico (la memoria usada por las estructuras de datos variables).

SIREN

Para una capa densa (completamente conectada) que transforma una entrada de dimensión n a una salida de dimensión m , el número de parámetros (pesos y sesgos) es $n \times m + m$.

En el contexto del modelo SIREN y asumiendo que cada parámetro se almacena como un `float32` (4 bytes), el coste espacial total es el siguiente

- La primera capa densa, que acepta la entrada del modelo $n = 3; m = 256$, tiene un coste de $(n \times m + m = 3 \times 256 + 256) \times 4 = 4096$ bytes.
- Cada capa intermedia donde $n = m = 256$ tiene un coste espacial es de $(n \times m + m = 256 \times 256 + 256) \times 4 = 263168$ bytes.
- La última capa densa que produce la salida del modelo $n = 256; m = 1$, tiene un coste de $(n \times m + m = 256 \times 1 + 1) \times 4 = 1028$ bytes.

El modelo debe almacenar un total de $(3 \times 256 + 256) + 2 \times (256 \times 256 + 256) + (256 \times 1 + 1) = 132865$ parámetros y su coste espacial es de $132865 \times 4 = 531460$ bytes = 519 kb. Si consideramos que el coste espacial de los gradientes es aproximadamente el mismo que el de los parámetros, entonces el coste total es de 1138 kb ≈ 1 Mb.

Características de Fourier

En el modelo con características de Fourier, el coste espacial se determina igualmente por el número total de parámetros, incluyendo pesos y sesgos, almacenados como float32 (4 bytes por parámetro).

- La primera capa densa, que acepta la entrada del modelo $n = 512; m = 256$, tiene un coste de $((n + m) \times m = 512 \times 256 + 512) \times 4 = 525312$ bytes.
- Cada capa intermedia donde $n = m = 256$ tiene un coste espacial es de $(n \times m + m = 256 \times 256 + 256) \times 4 = 263168$ bytes.
- La última capa densa que produce la salida del modelo $n = 256; m = 1$, tiene un coste de $(n \times m + m = 256 \times 1 + 1) \times 4 = 1028$ bytes.

El modelo debe almacenar un total de $(512 \times 256 + 256) + 2 \times (256 \times 256 + 256) + (256 \times 1 + 1) = 263169$ parámetros y su coste espacial es de $263169 \times 4 = 1052676$ bytes ≈ 1028 kb ≈ 1 Mb. Considerando, como antes, que el coste espacial de los gradientes es aproximadamente el mismo que el de los parámetros, el coste total es de 2056 kb ≈ 2 Mb. Estos cálculos nos permiten anticipar el crecimiento del coste espacial del modelo en función de la arquitectura de sus capas.

A.4. Rendimiento obtenido con *pytorch profiler*

PyTorch Profiler es una herramienta integrada en la biblioteca PyTorch que permite analizar y optimizar el rendimiento de las redes neuronales durante el entrenamiento. Proporciona un análisis del tiempo y del uso de los recursos en cada paso del proceso, ofreciendo información sobre el coste computacional de las diferentes operaciones de la red. Permite identificar cuellos de botella y otros problemas para optimizar el rendimiento de los modelos.

En este apartado, presentamos los datos de rendimiento obtenidos durante el entrenamiento de los experimentos 4.2.1 y 4.2.2, utilizando el modelo de GPU de gama más baja disponible, la

Tesla T4 y la de gama más alta, la A100. La visualización de estos datos se ha realizado con TensorBoard¹.

Comparación de rendimiento y tiempo de uso de la GPU

Los parámetros medidos en este apartado son los siguientes:

- **GPU Utilization:** Indica el porcentaje de tiempo en el que la GPU está realizando cálculos activamente.
- **Est. SM Efficiency:** Representa la eficiencia de los Multiprocesadores de Streaming (SM), que son los núcleos que ejecutan los kernels en la GPU.
- **Est. Achieved Occupancy:** Mide la eficiencia en el uso de los recursos de hardware en los multiprocesadores.
- **Average Step Time:** Tiempo promedio que toma un paso completo de entrenamiento, incluyendo el forward y backward pass y, en nuestro caso el tiempo de evaluación (test).
- **Kernel (%)**: Porcentaje de tiempo dedicado a la ejecución de los kernels en la GPU.
- **Memcpy (%)**: Tiempo empleado en copiar datos entre la memoria de la CPU y la GPU.
- **CPU Exec (%)**: Tiempo empleado en cálculos realizados en la CPU.

Parámetro	Fourier features $\sigma = [10, 10, .8]$ $k = 512$		SIREN $\omega = 50$	
	Tesla T4	NVIDIA A100	Tesla T4	NVIDIA A100
GPU				
Memory (GB)	14.75	39.56	14.75	39.56
Compute Capability	7.5	8.0	7.5	8.0
Number of Workers	4	4	4	4
GPU Utilization (%)	97.91	92.11	97.67	91.98
Est. SM Efficiency (%)	97.87	91.46	97.87	91.86
Est. Achieved Occupancy (%)	66.24	44.41	69.75	53.25
Average Step Time (μs)	2.359.548	499.543	1.994.978	402.831
Kernel (%)	97.91	92.11	97.67	91.98
Memcpy (%)	0.03	0.01	0.03	0.04
CPU Exec (%)	0.59	4.2	0.51	2.91

Tabla A.2: Comparación de parámetros entre Fourier features y SIREN con GPUs T4 y A100

Los resultados de la Tabla A.2 muestran que la GPU NVIDIA A100 es significativamente más rápida que la Tesla T4, con tiempos de paso promedio sustancialmente menores. Sin embargo, la ocupación lograda en la A100 es menor –quizá mejorarse con ajustes de los kernels

¹El resto de los datos obtenidos con PyTorch Profiler para cada experimento y para cada modelo de GPU (T4, L4, V100 y A100) se encuentran en el DVD que acompaña este proyecto, junto con las instrucciones para visualizarlos.

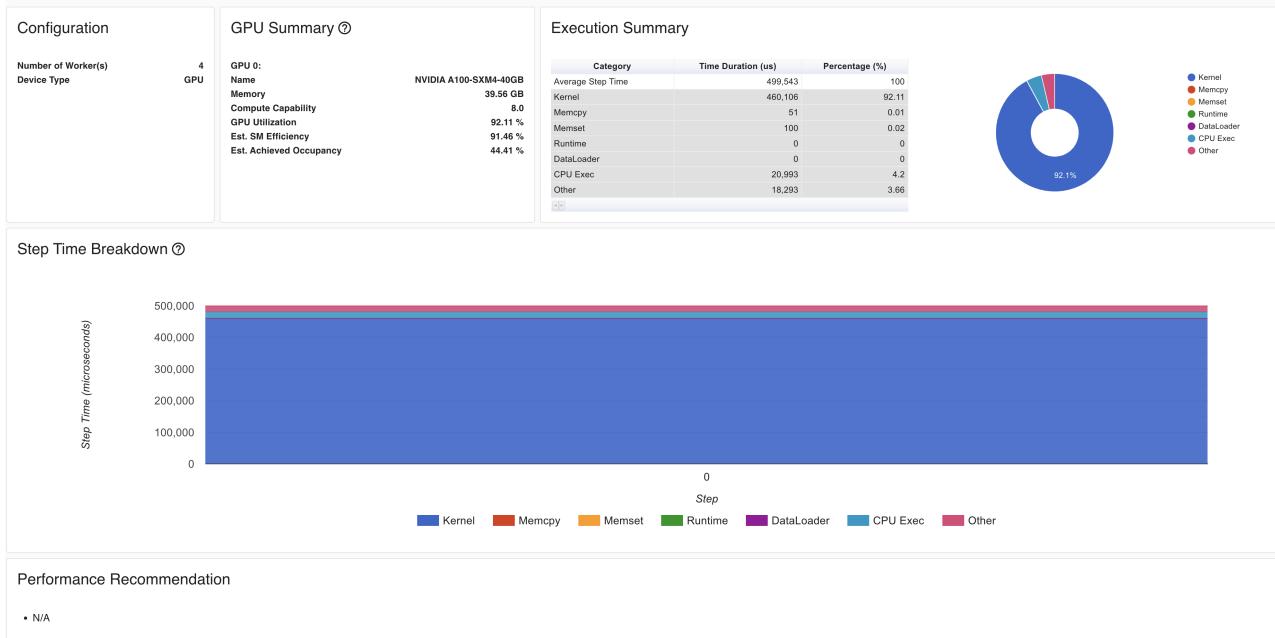


Figura A.1: Vista de tensorboard para el resumen de ejecución del modelo de Fourier features utilizando la GPU A100 en Google Colab.

y del tamaño del lote—. Por último, la eficiencia SM y el uso de los kernels siguen siendo altos en ambas GPUs, lo que indica un uso efectivo de los recursos. La Fig. A.1 muestra la vista con los datos generales obtenidos durante la ejecución del modelo de Fourier features con un $\sigma = [10, 10, .8]$ y $k = 512$ con la GPU A100.

Comparación de uso de Memoria

Se han comparado los modelos SIREN y Fourier features en GPUs T4 y A100, analizando su uso de memoria. Los parámetros más importantes se presentan a continuación:

Parámetro	Fourier features (T4)	Fourier features (A100)	SIREN (T4)	SIREN (A100)
Peak Memory Usage (MB)	5.297,1	5.294,1	6.400,9	18.915,4
Allocated Memory (MB)		Variable durante el entrenamiento		
Reserved Memory (MB)		Variable durante el entrenamiento		

Tabla A.3: Comparación de uso de memoria entre Fourier features y SIREN con GPUs T4 y A100

En la Tabla A.3 se observa que el mayor pico de memoria se produce durante el entrenamiento del modelo SIREN en la GPU A100, mientras que el modelo de Fourier features tiene un uso más moderado en ambas GPUs. Esto se debe a que para el entrenamiento del modelo SIREN se usó un tamaño de batch bastante mayor. La memoria asignada y reservada varían durante el entrenamiento debido a la gestión dinámica de memoria de PyTorch. La Fig. A.2 muestra esta variación en el modelo de Fourier features.

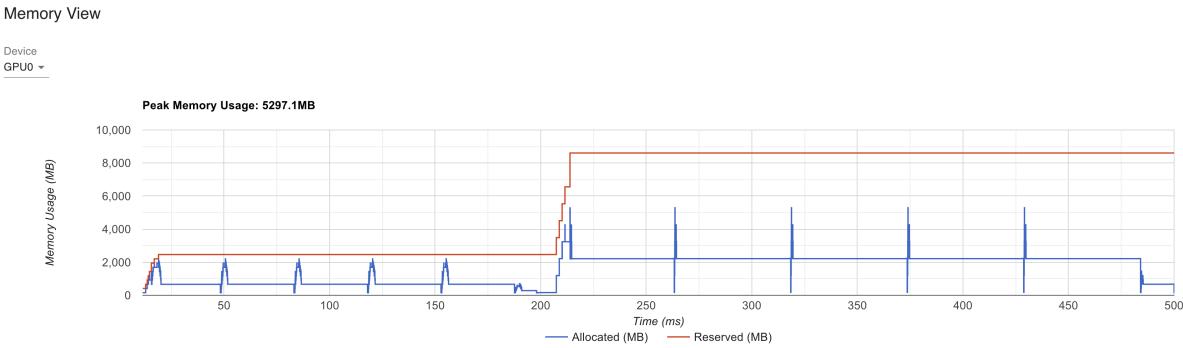


Figura A.2: Vista de tensorboard del uso de la memoria asignada y reservada del modelo de Fourier features utilizando la GPU A100 en Google Colab.