



UNIVERSIDAD
DE GRANADA

Facultad de Ciencias y Escuela Técnica Superior de Ingenierías
Informática y de Telecomunicación.

DOBLE GRADO EN MATEMÁTICAS E INGENIERÍA
INFORMÁTICA

TRABAJO DE FIN DE GRADO

Entrenamiento de Redes Convolucionales mediante la Transformada de Fourier

Isabel María Moreno Cuadrado

Curso académico 2023-2024

Entrenamiento de Redes Convolucionales mediante la Transformada de Fourier

Isabel María Moreno Cuadrado

Isabel María Moreno Cuadrado *Entrenamiento de Redes Convolucionales mediante la Transformada de Fourier*.

Trabajo de fin de Grado. Curso académico 2023-2024.

**Responsable de
tutorización**

Francisco Javier Meri de la Maza
Departamento de Análisis Matemático

Facultad de Ciencias

Jesús Giráldez Crú
*Departamento de Ciencias de la
Computación e Inteligencia Artificial*

Escuela Técnica Superior de
Ingenierías Informática y de
Telecomunicación

Doble Grado en
Matemáticas e Ingeniería
Informática

Universidad de Granada

DECLARACIÓN DE ORIGINALIDAD

D./Dña. Isabel María Moreno Cuadrado

Declaro explícitamente que el trabajo presentado como Trabajo de Fin de Grado (TFG), correspondiente al curso académico 2023-2024, es original, entendido esto en el sentido de que no he utilizado para la elaboración del trabajo fuentes sin citarlas debidamente.

En Granada, a 28 de septiembre de 2024.

Fdo: Isabel María Moreno Cuadrado

*A mis padres, por permitirme
alcanzar mi sueño:
Ser Matemática e Informática.*

Agradecimientos

Deseo aprovechar este espacio para expresar mi sincero agradecimiento a todas las personas que han sido fundamentales en este camino académico y personal:

A mis estimados tutores, Javier y Jesús, cuya orientación, apoyo y conocimiento han sido imprescindibles en cada etapa de este proceso.

A mis padres y mi hermana por su amor, paciencia y apoyo incondicional que han sido fundamentales en mi camino hacia el éxito.

A Lala y Lolo por que han sido mi fuente de fuerza y motivación para alcanzar todos mis objetivos.

A Kiko, que me ha enseñado que no hay que conformarse con menos de lo que uno vale y ha sabido sacar siempre lo mejor de mí.

A Natalia, Carmen, Ana y Bea porque nunca pasa el tiempo entre nosotras y siempre tengo un lugar seguro al que volver.

A mis compañeros, por hacerme sentir acogida y segura durante este camino de incertidumbre; lo cierto es que puedo decir que he tenido la inmensa suerte de graduarme con mis amigos.

A Armando Villena, por mostrarme la belleza que reside en el Análisis de Fourier, a Pablo Mesejo, por enseñarme a disfrutar de la Informática, a Nuria, por estar siempre dispuesta a ayudarme, incluso cuando me rodean los lobos. Y de nuevo, a Javier por despertarme el interés por la investigación.

Por último, a mí, por todos aquellos sacrificios que he tenido que hacer para llegar hasta este preciso instante y poder decir, con orgullo, que soy Matemática e Informática por la Universidad de Granada y en este documento reside la prueba de ello.

Entrenamiento de Redes Convolucionales mediante la Transformada de Fourier

Isabel María Moreno Cuadrado

Palabras clave:

Redes Neuronales Convolucionales (CNN), Transformada de Fourier, Convolución, Transformada Rápida de Fourier (FFT), Transformada de Fourier Discreta (DFT), Aprendizaje Profundo (DL), Visión por Computador (VC).

Resumen

En este TFG, estructurado en dos partes distintas, una de índole matemática y otra de naturaleza informática, se exploran cuestiones vinculadas al campo del DL, concretamente de las CNN, estableciendo una interconexión entre ambas áreas.

En la parte matemática de este TFG, correspondiente a la primera sección del mismo, se estudia en profundidad el análisis de Fourier en $\mathcal{L}^1(\mathbb{R}^n)$. El análisis de la Transformada de Fourier en este espacio, motivará posteriormente la definición de su versión en el marco discreto, la DFT, cuyas propiedades y características más importantes emergerán de manera natural y estarán inspiradas por su similitud con las propiedades descritas en el ámbito continuo. Adicionalmente, se presentará en esta primera parte, la operación de convolución, que junto con el Teorema de Convolución consolidarán la base teórica para el desarrollo de la segunda parte de esta memoria.

En la parte informática de este TFG, correspondiente a la segunda sección del mismo, se estudia un método alternativo para realizar la operación de convolución entre un núcleo y una imagen, usando el Teorema de Convolución y las propiedades de la DFT. Este método utiliza la FFT para convertir los productos matriciales entre ambos operandos en productos puntuales. En esta parte, se realizan, por tanto, una serie de experimentos con objeto de evaluar la viabilidad del nuevo método de convolución propuesto, en problemas de VC. Posteriormente, se analiza la eficiencia de este algoritmo, lo cual requiere un estudio detallado de la eficiencia del algoritmo de la FFT. Finalmente, este algoritmo se termina incorporando en la arquitectura de una CNN, estudiando una nueva metodología propuesta en los trabajos futuros de [10] para entrenar esta red íntegramente en el dominio de la frecuencia, y realizando posteriormente una comparativa de su rendimiento con el entrenamiento en una arquitectura clásica.

Esta aproximación permite entrenar una CNN usando menos operaciones y, por lo tanto, representa un avance prometedor en la línea de investigación; aceleración del entrenamiento de una CNN. Este avance es especialmente relevante dado el aumento en el número de mapas de características en las CNN modernas, que además suelen trabajar con datos de grandes dimensiones. Esta metodología se alinea con la creciente concienciación sobre la “Green AI”, que busca desarrollar tecnologías y técnicas de IA sostenibles, promoviendo un enfoque más eficiente y responsable.

Se muestra, por tanto, en el presente documento, cómo la parte matemática sustenta y motiva nuevos avances en una línea de investigación dentro del DL, conectando de esta manera ambas disciplinas y mostrando cuán fructíferos pueden ser los resultados cuando se combinan ambas.

Training Convolutional Networks Using the Fast Fourier Transform

Isabel María Moreno Cuadrado

Keywords:

Convolutional Neural Networks (CNN), Fourier Transform, Convolution, Fast Fourier Transform (FFT), Discrete Fourier Transform (DFT), Computer Vision (VC), Deep Learning (DL)

Abstract

In this undergraduate thesis, structured in two distinct parts, one of mathematical nature and the other one of computational nature, some issues related to the field of DL, specifically CNN, are explored, establishing an interconnection between both areas.

In the mathematical part of this mark, corresponding to the first section, the Fourier analysis in $\mathcal{L}^1(\mathbb{R}^n)$ is studied in depth. The analysis of the Fourier Transform in this space will subsequently motivate the definition of its version in the discrete framework, the DFT (Discrete Fourier Transform), whose most important properties and characteristics will emerge naturally and will be inspired by their similarity with the properties described in the continuous domain. Additionally, in this first part, the convolution operation will be presented, which, along with the Convolution Theorem, will consolidate the theoretical basis for the development of the second part of this document.

In the computational part of this thesis, corresponding to the second section, an alternative method for performing the convolution operation between a kernel and an image using the Convolution Theorem and the properties of the DFT is studied. This method uses the FFT to convert the matrix products between both operands into pointwise products. Therefore, in this part, a series of experiments are conducted to evaluate the feasibility of the proposed new convolution method in VC problems. Subsequently, the efficiency of this algorithm is analyzed, which requires a detailed study of the efficiency of the FFT algorithm. Finally, this algorithm is incorporated into the architecture of a CNN, studying a new methodology proposed in the future works of [10] to train this network entirely in the frequency domain and subsequently comparing its performance with training in a classical architecture.

This approach allows training a CNN using fewer operations and, therefore, represents a promising advancement in the line of research aimed at accelerating CNN training. This advancement is especially relevant given the increase in the number of feature maps in modern CNN, which also tend to work with large-scale data. This methodology aligns with the growing awareness of "Green AI," which seeks to develop sustainable AI technologies and techniques, promoting a more efficient and responsible approach.

Therefore, this document shows how the mathematical part supports and motivates new advances in a line of research within DL, thus connecting both disciplines and demonstrating how fruitful the results can be when both are combined.

Índice general

1. Introducción	1
I. Parte Matemática. Análisis de Fourier en $\mathcal{L}^1(\mathbb{R}^n)$	7
2. Introducción a la Parte Matemática	9
3. Preliminares	13
3.1. Espacios $\mathcal{L}^p(\mathbb{R}^n)$	13
3.2. Definiciones	15
3.3. Módulo de Continuidad	16
4. Transformada de Fourier en $\mathcal{L}^1(\mathbb{R}^n)$	19
4.1. Definición	19
4.2. Propiedades de la Transformada de Fourier	20
4.3. Permutación Integratoria de la Transformada	22
4.4. Magnitud de la Transformada de Fourier	22
4.5. Continuidad de la Transformada de Fourier	25
4.6. Transformada de Fourier y Derivación	25
4.7. Ejemplos	28
4.8. Teorema de Inversión	31
4.9. Teorema de Unicidad	32
4.10. Clase de Schwartz	33
5. Convolución	37
5.1. Definición	37
5.2. Teorema de Convolución	39
5.3. Propiedades de la Convolución	41
5.4. Teorema de Convolución de Young	43
5.5. Derivabilidad de la Convolución	46
5.6. Núcleos de Sumabilidad	49
5.7. Métodos de Sumación	57
6. Transformada de Fourier en $\mathcal{L}^2(\mathbb{R}^n)$	63
6.1. Definición	63
6.2. Permutación Integratoria de la Transformada	67
6.3. Propiedades	68
6.4. Fórmulas de Parseval	70
6.5. Convolución	71
6.6. Teorema de Inversión	73

II. Parte Informática. Aceleración de Redes Neuronales Convolucionales mediante Análisis en el Dominio de la Frecuencia	75
7. Introducción a la Parte Informática	77
7.1. Descripción del Problema Planteado	78
7.2. Objetivos	81
8. Planificación	83
8.1. Metodología del Diseño	83
8.2. Planificación Temporal	84
8.3. Planificación Económica	84
9. Estado del Arte	87
9.1. Aprendizaje Automático	87
9.2. Visión por Computador	90
9.3. Aprendizaje Profundo	91
9.4. Redes Neuronales Convolucionales	98
10. Fundamentos Teóricos	105
10.1. Conceptos Clave en el Entrenamiento de Redes	105
10.2. Protocolos de Validación Experimental	106
10.3. Técnicas para Mejorar el Entrenamiento	107
11. Transformada de Fourier Discreta	113
11.1. Motivación	113
11.2. Transformada de Fourier Discreta 1D	115
11.3. Transformada de Fourier Discreta 2D	117
11.4. Filtrado Lineal	121
11.5. Teorema de Convolución	126
12. Transformada Rápida de Fourier	131
12.1. Historia	131
12.2. Método Directo	133
12.3. Método FFT	133
12.4. Algoritmos FFT	138
13. Análisis de la Eficiencia	143
13.1. Método Alternativo de la Convolución	143
13.2. Entrenamiento en el Dominio de la Frecuencia	144
14. Análisis Experimental	149
14.1. Entorno de Desarrollo	149
14.2. Método Alternativo de la Convolución.	150
14.3. Entrenamiento de CNN en el Dominio de la Frecuencia	157

III. Conclusiones y Trabajos Futuros	173
15. Conclusiones y Trabajos Futuros	175
15.1. Objetivos Satisfechos	176
15.2. Trabajos Futuros y Comentarios	176
Bibliografía	179

Capítulo 1.

Introducción

Vivimos inmersos en un mundo repleto de señales que fluyen continuamente a nuestro alrededor, desde el murmullo constante de la ciudad hasta el sonido discreto de una canción de fondo, o la complejidad visual de un paisaje urbano. Cada uno de estos estímulos representa una rica fuente de datos que, aunque a menudo no somos plenamente conscientes de su profundidad, configuran nuestra percepción y nuestras interacciones con el entorno. En este mar de datos, el análisis de Fourier emerge como una herramienta teórica esencial para decodificar y comprender las complejidades inherentes a estas señales descritas como funciones.

Esencialmente, el análisis de Fourier, que recibe su nombre por el matemático Jean-Baptiste Joseph Fourier, persigue dos propósitos: el análisis o descomposición de una función en sus armónicos, y la síntesis o recomposición de la función a partir de estos.

En general, esta rama del análisis está sustentada por dos pilares. El primero es el constituido por los conceptos de las series de Fourier (en el contexto periódico) y de la Transformada de Fourier (en el contexto no periódico). El segundo es el concepto de convolución de dos funciones, en particular, el Teorema de Convolución, que desempeña un papel fundamental en este trabajo.

Con el propósito de estudiar y manipular las señales, en la primera parte del presente trabajo, dedicado a la parte matemática del TFG, se realizará un análisis exhaustivo del Análisis de Fourier en el contexto no periódico, dado que, en principio, pueden existir señales que no presenten periodicidad.

Concretamente, se trabajará en el espacio $\mathcal{L}^1(\mathbb{R}^n)$, a pesar de que la mayoría de aproximaciones de la Transformada de Fourier se llevan a cabo en el espacio de Schwartz $\mathcal{S}^1(\mathbb{R}^n)$. Esto se debe a que la Transformada de Fourier muestra un comportamiento particularmente favorable en este último. Sin embargo, el enfoque que se toma, al ser más general, proporciona un campo de aplicabilidad mayor y el reto añadido de identificar las condiciones bajo las cuales ciertas propiedades y resultados se mantienen válidos en $\mathcal{L}^1(\mathbb{R}^n)$. Este reto se ha manifestado de forma significativa en las áreas de derivación y aplicación de la Transformada de Fourier, obligando a un examen meticuloso de cómo estas operaciones interactúan dentro de este espacio más general.

Por lo tanto, en la primera parte de la memoria se introducirá el concepto de *Transformada de Fourier en $\mathcal{L}^1(\mathbb{R}^n)$* y se estudiarán propiedades de esta como la *continuidad*, la *acotación*, y el *Lema de Riemann-Lebesgue*. Se realizará un examen detallado de la relación entre la Transformada de Fourier y los *procesos de derivación*. Después, se presentarán y demostrarán

CAPÍTULO 1. INTRODUCCIÓN

dos teoremas: el *Teorema de Inversión*, que permitirá bajo ciertas condiciones recuperar la función a partir de su Transformada de Fourier, y el *Teorema de Unicidad*, que surgirá como consecuencia. Adicionalmente, se introducirá la transformada de Fourier en la *Clase de Schwartz*, donde se describe su comportamiento particularmente favorable, y se estudiará también esta, en el espacio $\mathcal{L}^2(\mathbb{R}^n)$, donde se analizará la relación que guarda con la definición previamente estudiada en $\mathcal{L}^1(\mathbb{R}^n)$. A lo largo de esta parte primera parte de la memoria se abordará la operación de convolución, junto con sus principales propiedades y finalmente se presentará el Teorema de Convolución.

Esta teoría se desarrolla íntegramente en un ámbito continuo. Sin embargo, existen señales en el mundo real inherentemente discretas. Un ejemplo claro de esto son las imágenes digitales, que no son más que señales visuales representadas por una matriz de píxeles con valores específicos. Adicionalmente, dado que los ordenadores solo pueden manejar sumas finitas y conjuntos de datos discretos, surge la imperiosa necesidad de definir en el marco discreto un objeto matemático análogo a la transformada de Fourier. Este objeto es la *DFT* [1], que se presenta en la segunda parte de este TFG junto con algunas de sus propiedades y características más importantes, las cuales emergen de manera natural y están motivadas por su similitud con las propiedades descritas en el ámbito continuo. En cierto sentido, se puede pensar que se proyectará la teoría descrita en $\mathcal{L}^1(\mathbb{R}^n)$ en el marco discreto.

Se prestará especial atención a la operación de convolución, ya que en el campo de la VC [2, 3, 4] esta operación es clave. Concretamente, es ampliamente usada en el procesamiento de imágenes, permitiendo la detección y realce de características específicas dentro de una imagen. Véase un ejemplo de la operación de convolución en la Figura 1.1.

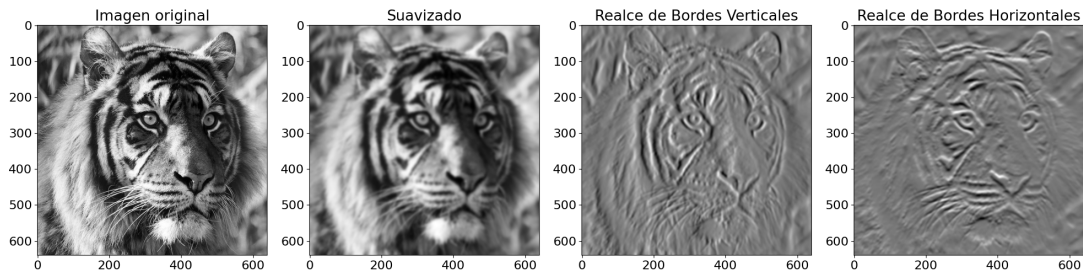


Figura 1.1.: De izquierda a derecha, se presenta la imagen original, la imagen convolucionada con un núcleo de suavizado, con uno de detección de bordes horizontales y de detección de bordes verticales.

Típicamente, la operación de convolución involucra una imagen de entrada y un determinado filtro o kernel, que es una matriz de dimensión reducida, diseñada para ser sensible a tipos particulares de características visuales. La operación de convolución, usando el *método tradicional*, implica deslizar el filtro o kernel sobre toda la imagen de entrada y calcular la suma de productos en cada posición del filtro sobre la imagen, generando una matriz de salida, conocida como mapa de características. La operación se describe en la Figura 1.2.

En esta segunda parte de la memoria se explorará un *algoritmo alternativo* al tradicional algoritmo de convolución, empleando el Teorema de Convolución. Este teorema afirma que la DFT de la convolución de una imagen I y un kernel K puede expresarse como el producto

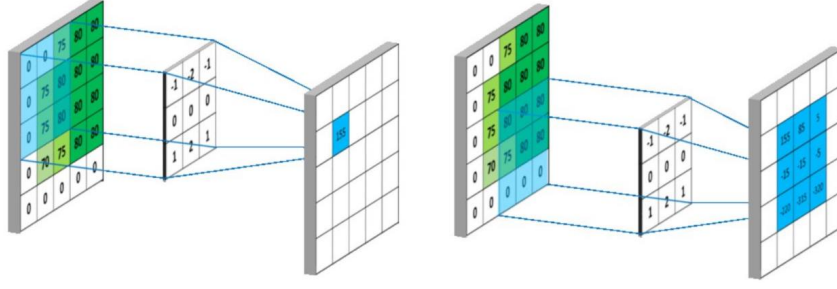


Figura 1.2.: Convolución de una imagen con un kernel de detección de bordes horizontales. Imagen obtenida de [5].

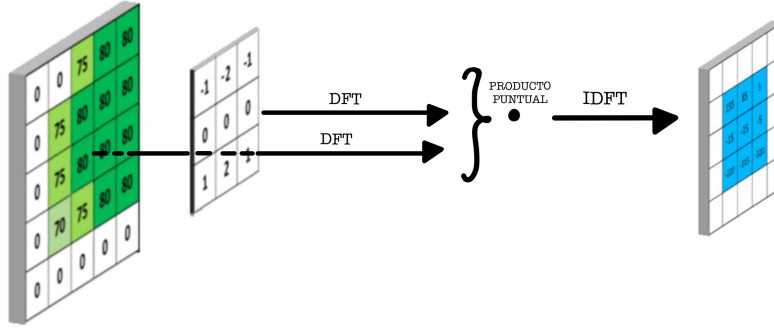


Figura 1.3.: Convolución de una imagen con un kernel de detección de bordes horizontales usando el Teorema de Convolución. Imagen modificada de [5].

puntual de las DFT de la imagen transformada $\hat{I}$ y del kernel transformado $\hat{K}$:

$$\widehat{I * K}(y) = \hat{I}(y) \cdot \hat{K}(y),$$

donde $*$ denota la operación de convolución. De este modo, al aplicar la *Transformada de Fourier Inversa* (*IDFT*), se puede recuperar la convolución entre la imagen y el núcleo, siendo por tanto la convolución en el dominio espacial equivalente en cierto sentido al producto en el dominio de la frecuencia (dominio de Fourier). Véase la Figura 1.3.

A pesar de que con este algoritmo propuesto las convoluciones se realizan como productos en el dominio de Fourier, cabe destacar que el algoritmo derivado del Teorema de Convolución requiere calcular la DFT de cada uno de los elementos involucrados en la operación de convolución y posteriormente aplicar la IDFT al resultado. Por tanto, la aplicación de estas transformadas puede ser computacionalmente costosa, limitando la utilidad del algoritmo. No obstante, el algoritmo de la *FFT* [6] permite el cálculo eficiente de la DFT. Por ello, para el estudio de la eficiencia del algoritmo de convolución propuesto, en esta segunda parte de la memoria se analizan las mejoras en la eficiencia que la FFT ofrece para calcular la DFT.

Este entendimiento avanzado del procesamiento de imágenes en el dominio de la frecuencia, gracias al Teorema de Convolución, también subyace en tecnologías más sofisticadas como las *CNN* [7], las cuales son modelos de Aprendizaje Automático (AA) [8, 9], especialmente importantes debido a su habilidad para procesar y analizar imágenes con una eficacia que

supera ampliamente a los métodos tradicionales.

Estas redes utilizan, como su nombre indica, la operación de convolución en su arquitectura, y entonces optimizar este algoritmo es importante para el rendimiento del modelo. Se plantea, por tanto como objetivo analizar una nueva metodología para el entrenamiento de CNN. Este estudio tiene como punto de partida el trabajo [10], que tomaremos como referencia, donde se sustituye la operación de convolución por la descrita en el Teorema de Convolución. Véase la Figura 1.4.

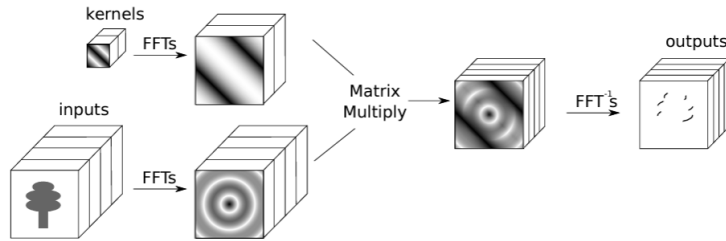


Figura 1.4.: Arquitectura CNN propuesta basada en la aplicación del Teorema de Convolución. Imagen adaptada de [10].

Lo cierto es que a pesar de que existen trabajos anteriores, como [11], en los que exploraron la posibilidad de utilizar la FFT para acelerar la inferencia en la primera capa de una red entrenada, no se llegó a utilizar durante el entrenamiento, posiblemente porque el número de mapas de características en ese momento era demasiado reducido como para que valiera la pena el esfuerzo de calcular las FFT en cada iteración. No obstante, dado el aumento en el número de mapas de características en las CNN modernas, resulta relevante estudiar esta línea de investigación.

En AA, inicialmente, los conjuntos de datos típicamente consistían en miles o decenas de miles de muestras [12, 13, 14]. Sin embargo, los conjuntos actuales incluyen millones de muestras [15, 16]. Este aumento considerable plantea retos importantes para entrenar modelos de forma eficiente en plazos aceptables. Incluso con el uso de computación paralela, entrenar una red en bases de datos como ImageNet [17] puede tomar semanas. Además, el consumo energético asociado con la Inteligencia Artificial (IA) es una preocupación creciente debido al incremento en la adopción y complejidad de estas tecnologías. Este consumo puede ser significativo, especialmente, en modelos de DL [7], que requieren extensos recursos computacionales para su entrenamiento y operación. En respuesta a estos desafíos, ha surgido el enfoque de la “Green AI”, que busca desarrollar tecnologías y técnicas de IA [18, 19] energéticamente eficientes y que minimicen su impacto ambiental. Reducir los tiempos de procesamiento es crucial para alcanzar los objetivos de la “Green AI”, ya que esto disminuye la energía necesaria para ejecutar estas tareas.

Esto es aún más importante en modelos como las CNN, que trabajan con imágenes, las cuales suelen ser datos de grandes dimensiones. Es por ello que esta línea de investigación es prometedora, y el presente trabajo se centra en estudiarla y proponer avances en ella.

Concretamente, la aproximación planteada en [10] todavía requiere aplicar la IDFT al resultado en cada convolución. Por este motivo, en la sección de trabajos futuros del artículo, se plantea la posibilidad de realizar el entrenamiento completamente en el dominio de Fourier, de modo que no sea necesario recurrir a la IDFT en cada etapa. Esta posibilidad es el avance propuesto

en este TFG, que se desarrollará en la segunda parte del trabajo, donde se presentará una CNN completamente entrenada en el dominio de la frecuencia, y se comparará con una arquitectura clásica.

El objetivo final de este trabajo es, por tanto, entrenar una CNN íntegramente en el dominio de la frecuencia utilizando el algoritmo de convolución propuesto que deriva del Teorema de Convolución. Para alcanzar este objetivo, es fundamental el análisis detallado de la DFT y la FFT, ya que ambos son componentes esenciales de dicho teorema. Para ello, la intuición y el enfoque derivados de la primera parte matemática, donde se desarrolla el análisis de Fourier en un contexto general, sirven para establecer una base teórica sólida que permita abordar de manera efectiva la implementación práctica del entrenamiento de CNN en el dominio de la frecuencia. Este enfoque teórico facilita la comprensión de cómo las señales pueden ser descompuestas y manipuladas en el dominio de la frecuencia, lo cual es crucial para aplicar correctamente el Teorema de Convolución.

El avance en esta línea de investigación no solo permitirá la posibilidad de estudiar el entrenamiento de las CNN de manera más eficiente y responsable, sino que también abrirá nuevas vías para el procesamiento de señales en diversas aplicaciones. Al dominar estos conceptos, se podrán desentrañar mejor las señales que forman parte de la vida cotidiana, desde imágenes médicas hasta datos de sensores en tiempo real, mejorando así nuestra capacidad para analizar y entender el mundo que nos rodea.

Parte I.

Parte Matemática. Análisis de Fourier en $\mathcal{L}^1(\mathbb{R}^n)$

Capítulo 2.

Introducción a la Parte Matemática

Uno de los propósitos principales de este trabajo es entrenar una CNN implementando las convoluciones como operaciones puntuales, con el fin de reducir el tiempo de entrenamiento y, por ende, contribuir directamente a una disminución del consumo energético. Reducir los tiempos de procesamiento es una preocupación creciente en la comunidad científica debido al incremento en la adopción y complejidad de los nuevos modelos de DL [7] que requieren extensos recursos computacionales para su entrenamiento y operación. La metodología alternativa para el entrenamiento de CNN que se propone en este trabajo, está basada en una aplicación del denominado *Teorema de Convolución* junto con el uso de la *FFT* [6], que permitirá el cálculo de la DFT en un tiempo aceptable.

El propósito de esta primera parte del TFG, dedicada a la contribución matemática, es desarrollar una base teórica sólida que permitirá establecer la definición de un objeto matemático de tal complejidad y riqueza que facilite no solo la derivación de la DFT a partir de este, sino también la comprensión intuitiva de sus propiedades esenciales, así como el Teorema de Convolución para el ámbito discreto.

La mayoría de los enfoques hacia la DFT surgen de manera natural a partir de las Series de Fourier. Sin embargo, se ha optado por desarrollar como punto de partida un marco teórico alternativo, que describimos a lo largo de este capítulo y que inspirará la definición de la DFT y de sus propiedades. Esta alternativa elegida se debe a la importancia de explorar diferentes interpretaciones, para así enriquecer nuestra comprensión de esta herramienta. Además, este enfoque permitirá estudiar de manera teórica ciertos aspectos específicos, como la derivación de la convolución, que son extensamente usados en el ámbito computacional.

Cabría pensar, entonces, que se introducirá la Transformada de Fourier en $\mathcal{L}^1(\mathbb{R})$. No obstante, optamos por centrar nuestro principal marco de estudio en $\mathcal{L}^1(\mathbb{R}^n)$, con el propósito de establecer un contexto más amplio y, así, poder efectuar más adelante una comparación con la DFT 2D (dos dimensiones), la cual es utilizada ampliamente en el campo de VC.

Finalmente, el lector se podría cuestionar la decisión de desarrollar la sección teórica en $\mathcal{L}^1(\mathbb{R}^n)$ en lugar de en el espacio de Schwartz $\mathcal{S}^1(\mathbb{R}^n)$, dado que la Transformada de Fourier muestra un comportamiento particularmente favorable en este último, y además, numerosos resultados se pueden obtener sin necesidad de hipótesis adicionales, más allá de que la función pertenezca a dicho espacio. De hecho, en la mayoría de fuentes bibliográficas consultadas [20, 21, 22], se introduce la Transformada de Fourier en $\mathcal{S}^1(\mathbb{R}^n)$. Sin embargo, se opta por desarrollar la teoría dentro del espacio $\mathcal{L}^1(\mathbb{R}^n)$ para ampliar significativamente nuestra capacidad de aplicar estos conceptos teóricos en un contexto más general. Esto ha planteado el reto añadido de identificar

las condiciones bajo las cuales ciertas propiedades y resultados se mantienen válidos en $\mathcal{L}^1(\mathbb{R}^n)$. Este reto se ha manifestado de forma significativa en las áreas de derivación y aplicación de la Transformada de Fourier, obligando a un examen meticuloso de cómo estas operaciones interactúan dentro de este espacio más general.

Describimos a continuación la estructura del resto de esta primera parte de la memoria, donde encontramos otros cuatro capítulos aparte de este, que permiten estructurar el contenido de manera coherente y detallada, abarcando desde los fundamentos teóricos hasta las aplicaciones prácticas específicas en el ámbito computacional. Cada capítulo se construye sobre el conocimiento previo, dando lugar a una comprensión profunda y gradual de la temática en estudio.

- En el segundo capítulo, **Preliminares**, se revisan los conceptos matemáticos y definiciones más relevantes, que serán usados con frecuencia más adelante en el texto. Cabe destacar el *módulo de continuidad de una función* que aparecerá de manera recurrente a lo largo del presente trabajo.
- El tercer capítulo, titulado **Transformada de Fourier en $\mathcal{L}^1(\mathbb{R}^n)$** , introduce la Transformada de Fourier en $\mathcal{L}^1(\mathbb{R}^n)$, así como sus propiedades más importantes. Dentro de este análisis, se profundiza en aspectos cruciales como la *continuidad*, la *acotación*, y el *Lema de Riemann-Lebesgue*. Además, se realiza un examen detallado de la relación entre la Transformada de Fourier y los *procesos de derivación*. Después, se presentan y demuestran dos teoremas: el *Teorema de Inversión*, que permitirá bajo ciertas condiciones recuperar la función a partir de su Transformada de Fourier, y el *Teorema de Unicidad*, que surgirá como consecuencia. Por último, se introduce la *Clase de Schwartz* y se describe el comportamiento particularmente favorable de la Transformada de Fourier en este espacio.
- Proseguimos con el cuarto capítulo titulado, **Convolución** y dedicado al estudio de esta operación esencial, junto con sus principales propiedades. Mediante el *Teorema de Young*, llevamos a cabo un análisis exhaustivo sobre las condiciones bajo las cuales está definida esta operación. Adicionalmente, introducimos el *Teorema de Convolución*, que constituye el núcleo de este trabajo. Posteriormente, estudiamos la *derivabilidad* de la convolución y cómo este resultado interactúa y repercute en el contexto computacional. Finalmente introducimos los *métodos de sumación* a partir de un estudio de los *núcleos de sumabilidad*, que permitirán construir otros mecanismos inversos de la transformada de Fourier.
- Finalizamos esta primera parte matemática de la memoria con el capítulo **Transformada de Fourier en $\mathcal{L}^2(\mathbb{R}^n)$** . Este nuevo marco teórico nos permitirá estudiar el concepto de la Transformada de Fourier $\hat{f}$ para cualquier función $f \in \mathcal{L}^2(\mathbb{R}^n)$, así como sus propiedades y relación con la definición previamente establecida en $\mathcal{L}^1(\mathbb{R}^n)$. Presentamos en este capítulo el *Teorema de Plancharel* y las *fórmulas de Parseval*. Además, exploraremos otras propiedades relevantes que posteriormente identificaremos en el contexto computacional.

En la elaboración de esta primera parte de la memoria, se ha recurrido principalmente a una serie de fuentes bibliográficas destacadas para cimentar las bases teóricas y prácticas presentadas. Entre éstas, sobresale el uso de los apuntes de la asignatura Análisis de Fourier en la Universidad de Granada, impartida por Armando Reyes Villena Muñoz. A pesar de tratar sobre la Transformada de Fourier en $\mathcal{L}^1(\mathbb{R})$, su claridad es tal que han facilitado la generalización de

muchos conceptos presentados a $\mathcal{L}^1(\mathbb{R}^n)$. Otros textos que han sido consultados con frecuencia incluyen el libro [22], que ha sido una fuente primordial para la demostración del Lema de Riemann-Lebesgue, además de proporcionar información detallada sobre $\mathcal{S}(\mathbb{R}^n)$. Adicionalmente, el texto [20] ha contribuido a una ampliación de conocimientos acerca de cómo la teoría se extiende de $\mathcal{L}^1(\mathbb{R})$ a $\mathcal{L}^1(\mathbb{R}^n)$. Como complemento, se han consultado otras referencias valiosas, tales como [23] y [24], para profundizar en diferentes aspectos y aplicaciones del Análisis de Fourier.

Capítulo 3.

Preliminares

De ahora en adelante trabajaremos con el espacio vectorial $\mathbb{R}^n$. En este espacio definimos el producto escalar de dos vectores $x = (x_1, x_2, \dots, x_n) \in \mathbb{R}^n$ e $y = (y_1, y_2, \dots, y_n) \in \mathbb{R}^n$ como el número real $\langle x, y \rangle$ dado por

$$\langle x, y \rangle = \sum_{k=1}^n x_k y_k. \quad (3.1)$$

El espacio $\mathbb{R}^n$ dotado del producto escalar definido en (3) es un espacio de Hilbert, que se conoce como el espacio euclídeo n -dimensional.

La norma asociada a dicho producto escalar es la norma euclídea $\|\cdot\|_2$. El espacio euclídeo n -dimensional es un espacio normado con esta norma. En $\mathbb{R}^n$ es posible considerar otras normas distintas. Sin embargo, en ausencia de especificación, se presume el uso de la norma euclídea.

El espacio $\mathbb{R}^n$ es además un espacio topológico con la topología usual que es la generada por la distancia asociada a la norma en $\mathbb{R}^n$. Es indiferente que norma se considere ya que todas ellas son equivalentes.

3.1. Espacios $\mathcal{L}^p(\mathbb{R}^n)$

Sea $\mathcal{L}^0(\mathbb{R}^n) = \{f : \mathbb{R}^n \rightarrow \mathbb{C} : f \text{ es medible}\}$. Para cada $p \in \mathbb{R}$ con $p \geq 1$, se define

$$\mathcal{L}^p(\mathbb{R}^n) = \left\{ f \in \mathcal{L}^0(\mathbb{R}^n) : \int_{\mathbb{R}^n} |f(x)|^p dx < \infty \right\},$$

y para cada función $f \in \mathcal{L}^p(\mathbb{R}^n)$, se define la seminorma

$$\|f\|_p = \left(\int_{\mathbb{R}^n} |f(x)|^p dx \right)^{\frac{1}{p}}.$$

Definimos también

$$\mathcal{L}^\infty(\mathbb{R}^n) = \{f \in \mathcal{L}^0(\mathbb{R}^n) : f \text{ está esencialmente acotada en } \mathbb{R}^n\},$$

CAPÍTULO 3. PRELIMINARES

y, dada una función $f \in \mathcal{L}^\infty(\mathbb{R}^n)$, se define

$$\|f\|_\infty = \inf\{M \in [0, \infty[: |f(x)| \leq M \text{ c.p.d. en } \mathbb{R}^n\}.$$

Para cada $1 \leq p \leq \infty$, el espacio $L^p(\mathbb{R}^n)$ es el espacio de Banach originado por la identificación en $\mathcal{L}^p(\mathbb{R}^n)$ de las funciones que coinciden casi por doquier. Por lo que los elementos de $L^p(\mathbb{R}^n)$ no son funciones sino clases de funciones bajo la relación de equivalencia de “ser iguales casi por doquier”.

El espacio $L^2(\mathbb{R}^n)$ es digno de atención especial, siendo un espacio de Hilbert cuyo producto escalar se define mediante la fórmula

$$\langle f, g \rangle = \int_{\mathbb{R}^n} f(x) \overline{g(x)} dx \quad \forall f, g \in L^2(\mathbb{R}^n).$$

Proposición 3.1.1. *Sea $f, g \in L^2(\mathbb{R}^n)$. Se verifica la identidad*

$$\langle f, g \rangle = \frac{1}{4} [\|f + g\|_2^2 - \|f - g\|_2^2 + i\|f + ig\|_2^2 - i\|f - ig\|_2^2].$$

denominada identidad de polarización.

Proposición 3.1.2. *Los espacios $\mathcal{L}^p(\mathbb{R}^n)$ son mutuamente incomparables.*

Demostración. Sea $1 \leq p < q \leq \infty$. Fijamos $p < a < q$ y definimos la siguiente función $f : \mathbb{R}^n \rightarrow \mathbb{R}$ como

$$f(x) = \begin{cases} \prod_{i=1}^n x_i^{-1/a} & \text{si } x_i > 1 \ \forall i \in \{1, \dots, n\}, \\ 0 & \text{en otro caso.} \end{cases}$$

Entonces se tiene que $f \notin \mathcal{L}^p(\mathbb{R}^n)$ y $f \in \mathcal{L}^q(\mathbb{R}^n)$.

Por otro lado definimos $g : \mathbb{R}^n \rightarrow \mathbb{R}$ como

$$f(x) = \begin{cases} \prod_{i=1}^n x_i^{-1/a} & \text{si } 0 < x_i < 1 \ \forall i \in \{1, \dots, n\}, \\ 0 & \text{en otro caso.} \end{cases}$$

Es claro que $f \notin \mathcal{L}^q(\mathbb{R}^n)$ y $f \in \mathcal{L}^p(\mathbb{R}^n)$.

Concluimos entonces que los conjuntos $\mathcal{L}^q(\mathbb{R}^n)$ y $\mathcal{L}^p(\mathbb{R}^n)$ son incomparables: $\mathcal{L}^q(\mathbb{R}^n) \not\subset \mathcal{L}^p(\mathbb{R}^n)$ y $\mathcal{L}^p(\mathbb{R}^n) \not\subset \mathcal{L}^q(\mathbb{R}^n)$. $\square$

Proposición 3.1.3. *Sea $f \in \mathcal{L}^1(\mathbb{R}^n)$. Entonces, para cada $t \in \mathbb{R}^n$, la función $x \mapsto f(x - t)$ es integrable en $\mathbb{R}^n$ y se cumple la identidad*

$$\int_{\mathbb{R}^n} f(x - t) dx = \int_{\mathbb{R}^n} f(x) dx.$$

Definimos otros dos espacios con los que también trabajaremos.

Definición 3.1.4. El conjunto $C_0(\mathbb{R}^n)$ es el conjunto de las funciones continuas $f : \mathbb{R}^n \rightarrow \mathbb{C}$ que se anulan en infinito.

Definición 3.1.5. El conjunto $C_{00}(\mathbb{R}^n)$ es el conjunto de las funciones continuas $f : \mathbb{R}^n \rightarrow \mathbb{C}$ cuyo soporte es compacto.

Proposición 3.1.6. $C_0(\mathbb{R}^n)$ es un espacio normado con la norma

$$\|f\|_\infty = \max\{|f(x)| : x \in \mathbb{R}^n\}, \quad \forall f \in C_0(\mathbb{R}^n).$$

y se tiene que $C_{00}(\mathbb{R}^n)$ es un subespacio denso en $C_0(\mathbb{R}^n)$.

Para $1 \leq p \leq \infty$, se tiene que $C_{00}(\mathbb{R}^n) \subset \mathcal{L}^p(\mathbb{R}^n)$. De hecho, sabemos que $C_{00}(\mathbb{R}^n)$ es denso en $\mathcal{L}^p(\mathbb{R}^n)$.

Teorema 3.1.7. Sea $1 \leq p < \infty$ y sea $f \in \mathcal{L}^p(\mathbb{R}^n)$. Entonces para cada $\epsilon \in \mathbb{R}^+$, existe $g \in C_{00}(\mathbb{R}^n)$ tal que $\|f - g\|_p < \epsilon$.

3.2. Definiciones

Definición 3.2.1. Sea $f \in \mathcal{L}^0(\mathbb{R}^n)$, para cada $a \in \mathbb{R}^+$, definimos $H_a f \in \mathcal{L}^0(\mathbb{R}^n)$ como $H_a f(x) = f(ax)$.

Proposición 3.2.2. Para $1 \leq p \leq \infty$, sea $f \in \mathcal{L}^p(\mathbb{R}^n)$. Entonces, para cada $a \in \mathbb{R}^+$, $H_a f \in \mathcal{L}^p(\mathbb{R}^n)$ y $\|H_a f\|_p = a^{-n/p} \|f\|_p$.

Demostración. Sea $1 \leq p < \infty$, y $a \in \mathbb{R}^+$. Usaremos que $f \in \mathcal{L}^p(\mathbb{R}^n)$. Entonces realizando un cambio de variable adecuado se tiene que

$$\int_{\mathbb{R}^n} |H_a f(x)|^p dx = \int_{\mathbb{R}^n} |f(ax)|^p dx = a^{-n} \int_{\mathbb{R}^n} |f(x)|^p dx = a^{-n} \|f\|_p^p < \infty.$$

De donde se deduce que $\|H_a f\|_p = a^{-n/p} \|f\|_p$.

Para $p = \infty$, se tiene que f está esencialmente acotada en $\mathbb{R}^n$. Entonces, como $|H_a f(x)|^p = |f(ax)|^p \forall x \in \mathbb{R}^n$, se tiene que $\tau_t f$ está esencialmente acotada en $\mathbb{R}^n$ y el supremo esencial coincide evidentemente en ambos casos. $\square$

Definición 3.2.3. Sea $f \in \mathcal{L}^0(\mathbb{R}^n)$. Para cada $t \in \mathbb{R}^n$, definimos

- $\tau_t f \in \mathcal{L}^0(\mathbb{R}^n)$ como $\tau_t f(x) = f(x - t) \forall x \in \mathbb{R}^n$,
- $\mu_t f \in \mathcal{L}^0(\mathbb{R}^n)$ como $\mu_t f(x) = e^{2\pi i \langle x, t \rangle} f(x) \forall x \in \mathbb{R}^n$.

Proposición 3.2.4. Para $1 \leq p \leq \infty$, sea $f \in \mathcal{L}^p(\mathbb{R}^n)$. Entonces, para cada $t \in \mathbb{R}^n$, $\tau_t f, \mu_t f \in \mathcal{L}^p(\mathbb{R}^n)$ y $\|\tau_t f\|_p = \|\mu_t f\|_p = \|f\|_p$.

Demostración. Sea $1 \leq p < \infty$, y $t \in \mathbb{R}^n$. Entonces

$$\begin{aligned} \int_{\mathbb{R}^n} |\tau_t f(x)|^p dx &= \int_{\mathbb{R}^n} |f(x - t)|^p dx = \int_{\mathbb{R}^n} |f(x)|^p dx = \|f\|_p^p < \infty, \\ \int_{\mathbb{R}^n} |\mu_t f(x)|^p dx &= \int_{\mathbb{R}^n} |e^{2\pi i \langle x, t \rangle} f(x)|^p dx = \int_{\mathbb{R}^n} |f(x)|^p dx = \|f\|_p^p < \infty. \end{aligned}$$

CAPÍTULO 3. PRELIMINARES

De donde se deduce que $\|\tau_t f\|_p = \|\mu_t f\|_p = \|f\|_p$.

Para $p = \infty$, se tiene que f está esencialmente acotada en $\mathbb{R}^n$. Entonces,

- Como $|\tau_t f(x)| = |f(x - t)| = |f(x)| \quad \forall x \in \mathbb{R}^n$, se tiene que $\tau_t f$ está esencialmente acotada en $\mathbb{R}^n$.
- Como $|\mu_t(x)| = |e^{2\pi i \langle x, t \rangle} f(x)| = |f(x)| \quad \forall x \in \mathbb{R}^n$, se sigue que $\mu_t f$ está esencialmente acotada en $\mathbb{R}^n$.

Además, el supremo esencial coincide evidentemente en los tres casos. $\square$

Definición 3.2.5. Sea $f \in \mathcal{L}^0(\mathbb{R}^n)$, definimos

- $\bar{f} \in \mathcal{L}^0(\mathbb{R}^n)$ como $\bar{f}(x) = \overline{f(x)} \quad \forall x \in \mathbb{R}^n$,
- $\tilde{f} \in \mathcal{L}^0(\mathbb{R}^n)$ como $\tilde{f}(x) = f(-x) \quad \forall x \in \mathbb{R}^n$.

Proposición 3.2.6. Para $1 \leq p \leq \infty$, sea $f \in \mathcal{L}^p(\mathbb{R}^n)$. Entonces, $\bar{f}, \tilde{f} \in \mathcal{L}^p(\mathbb{R}^n)$, y $\|\bar{f}\|_p = \|\tilde{f}\|_p = \|f\|_p$.

Demostración. Sea $1 \leq p < \infty$. Entonces

$$\begin{aligned} \int_{\mathbb{R}^n} |\bar{f}(x)|^p dx &= \int_{\mathbb{R}^n} |\overline{f(x)}|^p dx = \int_{\mathbb{R}^n} |f(x)|^p dx = \|f\|_p^p < \infty, \\ \int_{\mathbb{R}^n} |\tilde{f}(x)|^p dx &= \int_{\mathbb{R}^n} |f(-x)|^p dx = \|f\|_p^p < \infty. \end{aligned}$$

De donde se deduce que $\|\bar{f}\|_p = \|\tilde{f}\|_p = \|f\|_p$.

Para $p = \infty$, se tiene que f está esencialmente acotada en $\mathbb{R}^n$. Entonces,

- Como $|\tilde{f}(x)| = |f(-x)| \quad \forall x \in \mathbb{R}^n$, se tiene que $\tilde{f}$ está esencialmente acotada en $\mathbb{R}^n$.
- Como $|\bar{f}(x)| = |\overline{f(x)}| = |f(x)| \quad \forall x \in \mathbb{R}^n$, se sigue que $\bar{f}$ está esencialmente acotada en $\mathbb{R}^n$.

Además, el supremo esencial coincide evidentemente en los tres casos. $\square$

Estos resultados se usarán en adelante sin hacer mención explícita a ellos.

3.3. Módulo de Continuidad

Definición 3.3.1. Sea $1 \leq p < \infty$ y sea $f \in \mathcal{L}^p(\mathbb{R}^n)$. El *módulo de continuidad en media* de f es la función $w_p f : (\mathbb{R}^n) \rightarrow \mathbb{R}$ definida por

$$w_p f(t) = \|\tau_t f - f\|_p \quad \forall t \in \mathbb{R}^n.$$

Proposición 3.3.2. Sea $1 \leq p < \infty$ y sea $f \in \mathcal{L}^p(\mathbb{R}^n)$. Entonces,

- $w_p f(t) = w_p f(-t) \quad \forall t \in \mathbb{R}^n$,
- $0 \leq w_p f(t) \leq 2\|f\|_p \quad \forall t \in \mathbb{R}^n$,

3.3. MÓDULO DE CONTINUIDAD

- $w_p f$ es uniformemente continua en $\mathbb{R}^n$. En particular, $\lim_{t \rightarrow 0} w_p f(t) = 0$.

Demostración. Comenzamos probando la simetría de $w_p f$. Para ello, tomamos $t \in \mathbb{R}^n$, y observamos que:

$$w_p f(t) = \left(\int_{\mathbb{R}^n} |f(x-t) - f(x)|^p dx \right)^{\frac{1}{p}} = \left(\int_{\mathbb{R}^n} |f(y) - f(y+t)|^p dy \right)^{\frac{1}{p}} = w_p f(-t).$$

Continuamos observando que, en efecto, $0 \leq w_p f(t)$. Para la otra acotación, tomando $t \in \mathbb{R}^n$:

$$w_p f(t) = \|\tau_t f - f\|_p \leq 2\|f\|_p.$$

A continuación, probamos que $w_p f$ es uniformemente continua. Lo que haremos será utilizar el Teorema 3.1.7.

Dado $\epsilon \in \mathbb{R}^+$, existe una función g continua con soporte compacto tal que $\|f - g\|_p < \frac{\epsilon}{3}$.

Como la función g es continua con soporte compacto, sabemos que existe $R > 0$, tal que $\{x \in \mathbb{R}^n : g(x) \neq 0\} \subset B_{\|\cdot\|_\infty}(0, R)$. Además, al ser g uniformemente continua, existe $0 < \delta < 1$ tal que

$$x, y \in \mathbb{R}^n, |x - y| < \delta \implies |g(x) - g(y)| < \frac{\epsilon}{3M^{\frac{1}{p}}}, \quad (3.2)$$

con $M = \mu(B_{\|\cdot\|_\infty}(0, R+1))$ donde μ es la medida de Lebesgue.

Para cada $s, t \in \mathbb{R}^n$, tenemos que

$$|w_p f(s) - w_p f(t)| \leq \|(\tau_s f - f) - (\tau_t f - f)\|_p = \|\tau_s f - \tau_t f\|_p.$$

Usando la función g podemos escribir

$$\begin{aligned} \|\tau_s f - \tau_t f\|_p &= \|\tau_s f - \tau_t f + g - g + \tau_s g - \tau_s g + \tau_t g - \tau_t f\| \\ &= \|\tau_s(f - g) + (\tau_s g - \tau_t g) - \tau_t(f - g)\|_p \leq 2\|f - g\|_p + \|\tau_s g - \tau_t g\|_p. \end{aligned}$$

Tratamos entonces de acotar el segundo término. Tomamos $|s - t| < \delta$. Usando (3.3) y teniendo en cuenta que $g(x - (s - t)) = g(x) = 0 \ \forall x \notin B_{\|\cdot\|_\infty}(0, R+1)$, se tiene que

$$\|\tau_s g - \tau_t g\|_p = \left(\int_{\mathbb{R}^n} |g(x - (s - t)) - g(x)|^p dx \right)^{\frac{1}{p}} \leq \left(\int_{B_{\|\cdot\|_\infty}(0, R+1)} \left| \frac{\epsilon}{3M^{\frac{1}{p}}} \right|^p dx \right)^{\frac{1}{p}} \leq \frac{\epsilon}{3}.$$

Deducimos finalmente que

$$|w_p f(s) - w_p f(t)| \leq 2\|f - g\|_p + \|\tau_s g - \tau_t g\|_p \leq \epsilon.$$

Luego $w_p f$ es uniformemente continua en $\mathbb{R}^n$. Si tomamos $\lim_{t \rightarrow 0} w_p f(t) = w_p f(0) = 0$ obtenemos la última observación.

□

Capítulo 4.

Transformada de Fourier en $\mathcal{L}^1(\mathbb{R}^n)$

4.1. Definición

Como se examinará en esta sección, existen múltiples enfoques para establecer la definición de la Transformada de Fourier en $\mathbb{R}^n$. La formulación que adoptaremos se basa en las presentadas en las referencias [24, 23]. No obstante, se considerarán y discutirán otras definiciones alternativas, como las propuestas en [20, 21].

Definición 4.1.1. Sea $f \in \mathcal{L}^1(\mathbb{R}^n)$. La Transformada de Fourier de f es la función $\widehat{f} : \mathbb{R}^n \rightarrow \mathbb{C}$ definida por

$$\widehat{f}(y) = \int_{\mathbb{R}^n} f(x) e^{-2\pi i \langle x, y \rangle} dx \quad \forall y \in \mathbb{R}^n.$$

Observación 4.1.2. Lo primero que debemos mencionar es que la definición anterior es legítima: dado $y \in \mathbb{R}^n$, la función $x \mapsto f(x) e^{-2\pi i \langle x, y \rangle}$ es medible por ser el producto de una función medible f y la función continua $x \mapsto e^{-2\pi i \langle x, y \rangle}$, y además

$$\int_{\mathbb{R}^n} |f(x) e^{-2\pi i \langle x, y \rangle}| dx = \int_{\mathbb{R}^n} |f(x)| dx < \infty,$$

luego es integrable en $\mathbb{R}^n$ para cada $y \in \mathbb{R}^n$.

Observación 4.1.3. Como el lector puede imaginar, uno de nuestros objetivos es ser capaces de reconstruir la función f a partir de $\widehat{f}$. Esta tarea se pretende llevar a cabo mediante la fórmula

$$f(x) = \int_{\mathbb{R}^n} \widehat{f}(y) e^{2\pi i \langle x, y \rangle} dy.$$

Esto implica que la función Transformada $\widehat{f}$ debe ser integrable sobre $\mathbb{R}^n$, lo cual puede no ser siempre el caso, incluso para funciones aparentemente simples. En situaciones donde la función Transformada no cumpla con el requisito de ser integrable, se consideran otras interpretaciones alternativas a la fórmula propuesta.

Observación 4.1.4. Es interesante observar que el concepto de la Transformada de Fourier tiene sentido para funciones definidas casi por doquier en $\mathbb{R}^n$. Además, si $f = g$ c.p.d. en $\mathbb{R}^n$, entonces se tiene que $\widehat{f} = \widehat{g}$. Por tanto, la Transformada de Fourier se transfiere naturalmente al espacio $L^1(\mathbb{R}^n)$.

Observación 4.1.5. Es frecuente encontrar otros textos en los que la transformada se define

mediante expresiones diferentes, como puede ser:

$$\int_{\mathbb{R}^n} f(x) e^{-i\langle x, y \rangle} dx,$$

definición que aparece en [20], y es aparentemente más sencilla. Lo cierto es que esto obliga a que la reconstrucción se haga mediante la fórmula

$$\frac{1}{(2\pi)^n} \int_{\mathbb{R}^n} f(x) e^{i\langle x, y \rangle} dy,$$

por lo que la constante (ligada a 2π) aparece en la reconstrucción, esta vez dependiente del número de dimensiones.

En general, si definimos la Transformada de Fourier como

$$a \int_{\mathbb{R}^n} f(x) e^{-i\langle x, y \rangle} dx,$$

esto obliga a que la reconstrucción venga dada por

$$b \int_{\mathbb{R}^n} f(x) e^{i\langle x, y \rangle} dy,$$

donde $a \cdot b = \frac{1}{(2\pi)^n}$.

Por tanto, no podemos eludir la presencia de la constante 2π en ninguna instancia. Sin embargo, parece que con nuestra elección minimizamos el riesgo de cometer un error, ya que al incorporar la constante en la exponencial, obtenemos fórmulas completamente simétricas y válidas para cualquier dimensión. Pero si el lector aún no ha quedado convencido, es crucial señalar que, usando la definición (4.1.1), el Teorema de Convolución (que será el núcleo central de este trabajo) no involucrará la presencia de constantes añadidas (aunque estas constantes sí aparecerán en el contexto de la Transformada de Fourier y la derivación).

4.2. Propiedades de la Transformada de Fourier

Demostraremos algunas propiedades de la Transformada de Fourier para comenzar a familiarizarnos con la definición.

Proposición 4.2.1. Sean $f, g \in \mathcal{L}^1(\mathbb{R}^n)$ y sean $\alpha, \beta \in \mathbb{C}$. Entonces

$$\widehat{\alpha f + \beta g} = \alpha \widehat{f} + \beta \widehat{g}.$$

Demostración. Para cada $y \in \mathbb{R}^n$, se verifica que

$$\begin{aligned} \widehat{\alpha f + \beta g} &= \int_{\mathbb{R}^n} (\alpha f(x) + \beta g(x)) e^{-2\pi i \langle x, y \rangle} dx \\ &= \alpha \int_{\mathbb{R}^n} f(x) e^{-2\pi i \langle x, y \rangle} dx + \beta \int_{\mathbb{R}^n} g(x) e^{-2\pi i \langle x, y \rangle} dx = \alpha \widehat{f} + \beta \widehat{g}. \end{aligned} \quad \square$$

Proposición 4.2.2. Sean $f, g \in \mathcal{L}^1(\mathbb{R}^n)$. Entonces

1. $\widehat{\widehat{f}}(y) = \widehat{f(-y)}$,
2. $\widehat{\widehat{\widehat{f}}}(y) = \widehat{f(-y)}$,
3. $\widehat{\widehat{\widehat{\widehat{f}}}}(y) = \widehat{f(y)}$.

Demostración.

$$\begin{aligned}\widehat{\widehat{f}}(y) &= \int_{\mathbb{R}^n} \overline{f(x)} e^{-2\pi i \langle x, y \rangle} dx = \int_{\mathbb{R}^n} \overline{f(x) e^{2\pi i \langle x, y \rangle}} dx = \overline{\int_{\mathbb{R}^n} f(x) e^{2\pi i \langle x, y \rangle} dx} = \widehat{f(-y)}, \\ \widehat{\widehat{\widehat{f}}}(y) &= \int_{\mathbb{R}^n} f(-x) e^{-2\pi i \langle x, y \rangle} dx = \int_{\mathbb{R}^n} f(x) e^{-2\pi i \langle (-x), y \rangle} dx = \int_{\mathbb{R}^n} f(x) e^{2\pi i \langle x, (-y) \rangle} dx = \widehat{f(-y)}, \\ \widehat{\widehat{\widehat{\widehat{f}}}}(y) &= \int_{\mathbb{R}^n} \overline{f(-x)} e^{-2\pi i \langle x, y \rangle} dx = \int_{\mathbb{R}^n} \overline{f(-x) e^{2\pi i \langle x, y \rangle}} dx = \overline{\int_{\mathbb{R}^n} f(x) e^{-2\pi i \langle x, y \rangle} dx} = \widehat{f(y)}. \quad \square\end{aligned}$$

Las siguientes proposiciones son útiles en lo que se refiere al cálculo de Transformadas de Fourier. Las demostraciones se reducen a efectuar un cambio de variable adecuado en la integral.

Proposición 4.2.3. Sea $f \in \mathcal{L}^1(\mathbb{R}^n)$ y sea $a \in \mathbb{R}^+$. Entonces, para cada $y \in \mathbb{R}^n$, se tiene:

$$\widehat{H_a f}(y) = a^{-n} \widehat{f}(a^{-1}y).$$

Demostración.

$$\widehat{H_a f}(y) = \int_{\mathbb{R}^n} f(ax) e^{-2\pi i \langle x, y \rangle} dx = \frac{1}{a^n} \int_{\mathbb{R}^n} f(x) e^{-2\pi i \langle x, \frac{y}{a} \rangle} dx = a^{-n} \widehat{f}(a^{-1}y). \quad \square$$

Proposición 4.2.4. Sea $f \in \mathcal{L}^1(\mathbb{R}^n)$ y sea $t \in \mathbb{R}^n$. Entonces, para cada $y \in \mathbb{R}^n$, se tiene:

$$\widehat{\tau_t f}(y) = e^{-2\pi i \langle t, y \rangle} \widehat{f}(y).$$

Demostración.

$$\begin{aligned}\widehat{\tau_t f}(y) &= \int_{\mathbb{R}^n} f(x-t) e^{-2\pi i \langle x, y \rangle} dx = \int_{\mathbb{R}^n} f(x) e^{-2\pi i \langle (x+t), y \rangle} dx \\ &= \int_{\mathbb{R}^n} f(x) e^{-2\pi i \langle x, y \rangle} e^{-2\pi i \langle t, y \rangle} dx = e^{-2\pi i \langle t, y \rangle} \widehat{f}(y). \quad \square\end{aligned}$$

Proposición 4.2.5. Sea $f \in \mathcal{L}^1(\mathbb{R}^n)$ y sea $t \in \mathbb{R}^n$. Entonces, para cada $y \in \mathbb{R}^n$, se tiene:

$$\widehat{\mu_t f}(y) = \widehat{f}(y-t).$$

Demostración.

$$\widehat{\mu_t f}(y) = \int_{\mathbb{R}^n} e^{2\pi i \langle t, x \rangle} f(x) e^{-2\pi i \langle x, y \rangle} dx = \int_{\mathbb{R}^n} f(x) e^{-2\pi i \langle x, (y-t) \rangle} dx = \widehat{f}(y-t). \quad \square$$

Proposición 4.2.6. Si $f \in \mathcal{L}^1(\mathbb{R}^n)$ es tal que $f(x_1, \dots, x_n) = f_1(x_1) \cdot f_2(x_2) \cdots f_n(x_n)$ $\forall (x_1, x_2, \dots, x_n) \in \mathbb{R}^n$, con $f_1, f_2, \dots, f_n \in \mathcal{L}^1(\mathbb{R})$. Entonces

$$\widehat{f}(y_1, y_2, \dots, y_n) = \widehat{f}_1(y_1) \cdot \widehat{f}_2(y_2) \cdots \widehat{f}_n(y_n) \quad \forall y = (y_1, y_2, \dots, y_n) \in \mathbb{R}^n.$$

Demostración. Sea $y = (y_1, y_2, \dots, y_n) \in \mathbb{R}^n$. Entonces, se tiene que

$$\begin{aligned} \widehat{f}(y) &= \int_{\mathbb{R}^n} (f_1(x_1)e^{-2\pi i(x_1 y_1)} \cdot f_2(x_2)e^{-2\pi i(x_2 y_2)} \cdots f_n(x_n)e^{-2\pi i(x_n y_n)}) dx_1 dx_2 \cdots dx_n \\ &= \prod_{j=1}^n \int_{\mathbb{R}} f_j(x_j)e^{-2\pi i x_j y_j} dx_j = \prod_{j=1}^n \widehat{f}_j(y_j). \end{aligned} \quad \square$$

Observación 4.2.7. Esta proposición nos permite reducir el cálculo de la Transformada de Fourier de una función integrable en $\mathbb{R}^n$ al caso $n = 1$ en determinadas situaciones.

4.3. Permutación Integratoria de la Transformada

El siguiente teorema nos permite intercambiar la Transformada dentro de la integral. Esta técnica resulta útil como estrategia para abordar ciertas demostraciones.

Teorema 4.3.1. Sean $f, g \in \mathcal{L}^1(\mathbb{R}^n)$. Entonces,

$$\int_{\mathbb{R}^n} \widehat{f}(x)g(x) dx = \int_{\mathbb{R}^n} f(x)\widehat{g}(x) dx.$$

Demostración. Por un lado tenemos que la función $(x, y) \mapsto f(x)g(y)e^{-2\pi i \langle x, y \rangle}$ es medible, y por otro se tiene que

$$\begin{aligned} &\int_{\mathbb{R}^n} \int_{\mathbb{R}^n} |f(x)g(y)e^{-2\pi i \langle x, y \rangle}| dx dy = \\ &= \int_{\mathbb{R}^n} \int_{\mathbb{R}^n} |f(x)g(y)| dx dy = \int_{\mathbb{R}^n} |f(x)| dx \int_{\mathbb{R}^n} |g(y)| dy < \infty, \end{aligned}$$

Por lo que el Teorema de Fubini-Tonelli asegura que la función es integrable en $\mathbb{R}^{n+n}$ y

$$\begin{aligned} \int_{\mathbb{R}^n} \widehat{f}(x)g(x) dx &= \int_{\mathbb{R}^n} \int_{\mathbb{R}^n} f(x)g(y)e^{-2\pi i \langle x, y \rangle} dy dx \\ &= \int_{\mathbb{R}^n} \int_{\mathbb{R}^n} f(x)g(y)e^{-2\pi i \langle x, y \rangle} dx dy = \int_{\mathbb{R}^n} f(x)\widehat{g}(x) dx. \quad \square \end{aligned}$$

4.4. Magnitud de la Transformada de Fourier

En esta sección estudiaremos la magnitud de la Transformada de Fourier, proporcionaremos una acotación de esta y demostraremos el Lema de Riemann-Lebesgue.

Teorema 4.4.1. Sea $f \in \mathcal{L}^1(\mathbb{R}^n)$. Entonces

$$\|\widehat{f}\|_{L^\infty} \leq \|f\|_{L^1}.$$

Demostración. Para cada $y \in \mathbb{R}^n$, se tiene que

$$|\widehat{f}(y)| = \left| \int_{\mathbb{R}^n} f(x) e^{-2\pi i \langle x, y \rangle} dx \right| \leq \int_{\mathbb{R}^n} |f(x) e^{-2\pi i \langle x, y \rangle}| dx = \int_{\mathbb{R}^n} |f(x)| dx = \|f\|_1.$$

De aquí se deduce que $\|\widehat{f}\|_{L^\infty} = \sup_{y \in \mathbb{R}^n} |\widehat{f}(y)| \leq \|f\|_1$. □

Teorema 4.4.2 (Lema de Riemann-Lebesgue). Sea $f \in \mathcal{L}^1(\mathbb{R}^n)$. Entonces,

$$\lim_{\|y\| \rightarrow \infty} \widehat{f}(y) = 0.$$

Antes de realizar la prueba recordamos algunas definiciones y resultados conocidos.

Definición 4.4.3. Un intervalo en $\mathbb{R}^n$ será un producto cartesiano de intervalos en $\mathbb{R}$ y denotaremos por $\mathcal{J}$ al conjunto de todos los intervalos acotados en $\mathbb{R}^n$.

Definición 4.4.4. Llamaremos función escalonada a toda combinación lineal de funciones características de intervalos acotados, es decir, a toda función $h : \mathbb{R}^n \rightarrow \mathbb{C}$ de la forma

$$h = \sum_{k=1}^n \alpha_k \chi_{J_k} \quad \text{donde} \quad n \in \mathbb{N}, \quad \alpha_1, \alpha_2, \dots, \alpha_n \in \mathbb{C}, \quad J_1, J_2, \dots, J_n \in \mathcal{J}.$$

Observación 4.4.5. El conjunto formado por todas las clases de equivalencia que contienen una función escalonada en $\mathbb{R}^n$ es denso en $L^1(\mathbb{R}^n)$. Como consecuencia, para cada $f \in \mathcal{L}^1(\mathbb{R}^n)$, existe una sucesión $\{g_n\}$ de funciones escalonadas que verifica:

$$\{g_n(x)\} \rightarrow f(x) \text{ p.c.t. } x \in \mathbb{R}^n, \quad \lim_{n \rightarrow \infty} \int_{\mathbb{R}^n} |f - g_n| dx = 0.$$

Demostración del Lema de Riemann-Lebesgue. La prueba está dividida en dos partes: primero se demuestra el teorema para cualquier función escalonada, y posteriormente, usando un argumento de densidad, se extiende el resultado a cualquier función integrable en $\mathbb{R}^n$.

Dado un intervalo $[a, b]$ en $\mathbb{R}$, consideramos la función característica $\chi_{[a, b]}$. Calculamos la Transformada de dicha función.

$$\begin{aligned} \widehat{\chi_{[a, b]}}(0) &= \int_{\mathbb{R}} \chi_{[a, b]} dx = b - a, \\ \widehat{\chi_{[a, b]}}(y) &= \int_{\mathbb{R}} \chi_{[a, b]} e^{-2\pi i xy} dx = \int_a^b e^{-2\pi i xy} dx = \frac{e^{-2\pi i ya} - e^{-2\pi i yb}}{2\pi i y} \quad \forall y \neq 0. \end{aligned}$$

Observamos que

$$\lim_{|y| \rightarrow \infty} \widehat{\chi_{[a, b]}}(y) = 0.$$

CAPÍTULO 4. TRANSFORMADA DE FOURIER EN $\mathcal{L}^1(\mathbb{R}^n)$

Tomamos ahora un intervalo en $\mathbb{R}^n$, $J = [a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_n, b_n]$ y la función característica χ_J . Procedemos calculando la Transformada de Fourier de $\chi_J \in \mathcal{L}^1(\mathbb{R}^n)$:

$$\begin{aligned}\widehat{\chi_J}(y_1, \dots, y_n) &= \int_{\mathbb{R}^n} \chi_J e^{-2\pi i \langle x, y \rangle} dx \\ &= \int_{a_n}^{b_n} \cdots \int_{a_2}^{b_2} \left(\int_{a_1}^{b_1} e^{-2\pi i x_1 y_1} dx_1 \right) e^{-2\pi i (x_2 y_2 + \dots + x_n y_n)} dx_2 \dots dx_n \\ &= \prod_{j=1}^n \left(\int_{a_j}^{b_j} e^{-2\pi i x_j y_j} dx_j \right).\end{aligned}$$

Sea un elemento $y = (y_1, y_2, \dots, y_n) \neq 0 \in \mathbb{R}^n$. Existe $j_0 \in \{1, \dots, n\}$ tal que $|y_{j_0}| > \frac{\|y\|}{\sqrt{n}}$. Notemos que, para todo $j \in \{1, \dots, n\}$, se tiene

$$\left| \int_{a_j}^{b_j} e^{-2\pi i x y_j} dx \right| \leq \int_{a_j}^{b_j} |e^{-2\pi i x y_j}| dx = (b_j - a_j).$$

Por otro lado,

$$\left| \int_{a_{j_0}}^{b_{j_0}} e^{-2\pi i x y_{j_0}} dx \right| = \left| \frac{e^{-2\pi i y_{j_0} a_{j_0}} - e^{-2\pi i y_{j_0} b_{j_0}}}{2\pi i y_{j_0}} \right| \leq \frac{2}{|2\pi i y_{j_0}|} < \frac{\sqrt{n}}{\pi \|y\|}.$$

Por tanto, tenemos que

$$|\widehat{\chi_J}(y)| \leq \prod_{\substack{j=1 \\ j \neq j_0}}^n (b_j - a_j) \cdot \frac{\sqrt{n}}{\pi \|y\|} \leq \max_{1 \leq j_0 \leq n} \prod_{\substack{j=1 \\ j \neq j_0}}^n (b_j - a_j) \cdot \frac{\sqrt{n}}{\pi \|y\|} = c \cdot \frac{\sqrt{n}}{\pi \|y\|},$$

donde la constante c no depende de j_0 . Atendiendo a esta última expresión, deducimos que

$$\lim_{\|y\| \rightarrow \infty} \widehat{\chi_J}(y) = 0.$$

Dada una función escalonada $g : \mathbb{R}^n \rightarrow \mathbb{R}$, esta será combinación lineal finita de funciones del tipo χ_J . Podemos concluir entonces que, gracias a la aditividad de los límites, el teorema se cumple para g . Así, se tendrá que

$$\lim_{\|y\| \rightarrow \infty} \widehat{g}(y) = 0.$$

Generalizamos ya el resultado. Dada ahora una función $f \in \mathcal{L}^1(\mathbb{R}^n)$ y $\epsilon > 0$, existe una función escalonada $h \in \mathcal{L}^1(\mathbb{R}^n)$ de manera que $\|f - h\|_{L^1} < \frac{\epsilon}{2}$. Por tanto, por lo probado anteriormente, existe una constante positiva M tal que si $\|y\| > M$ entonces $|\widehat{h}(y)| < \frac{\epsilon}{2}$. Luego

$$|\widehat{f}(y)| \leq |\widehat{f}(y) - \widehat{h}(y)| + |\widehat{h}(y)| \leq \|f - h\|_{L^1} + \frac{\epsilon}{2} \leq \epsilon,$$

para $\|y\| > M$, de donde se deduce que $\lim_{\|y\| \rightarrow \infty} \widehat{f}(y) = 0$, y hemos terminado. $\square$

4.5. Continuidad de la Transformada de Fourier

A continuación se describe la continuidad de la Transformada de Fourier $\hat{f}$ dada $f \in \mathcal{L}^1(\mathbb{R}^n)$.

Teorema 4.5.1. *Sea $f \in \mathcal{L}^1(\mathbb{R}^n)$. Entonces la función $\hat{f}$ es continua en $\mathbb{R}^n$.*

Demostración. Podemos tratar la integral que aparece en 4.1.1 como una integral dependiente de un parámetro. Sea $\gamma : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{C}$ la función dada por

$$\gamma(x, y) = f(x)e^{-2\pi i \langle x, y \rangle}.$$

Se cumple:

1. Para cada $y \in \mathbb{R}^n$, la función $x \mapsto \gamma(x, y)$ es integrable en $\mathbb{R}^n$,
2. Para cada $x \in \mathbb{R}^n$, la función $y \mapsto \gamma(x, y)$ es continua en $\mathbb{R}^n$,
3. $|\gamma(x, y)| = |f(x)e^{-2\pi i \langle x, y \rangle}| = |f(x)| \quad \forall x, y \in \mathbb{R}^n$.

Puesto que $|f|$ es integrable en $\mathbb{R}^n$, el Teorema de continuidad de las integrales dependientes de un parámetro garantiza que la función $\hat{f}$ es continua en $\mathbb{R}^n$. $\square$

Observación 4.5.2. Los Teoremas 4.5.1 y 4.4.2 aseguran que $\hat{f}$ es uniformemente continua en $\mathbb{R}^n$.

Observación 4.5.3. Los Teoremas 4.5.1 y 4.4.2 aseguran que $\hat{f} \in C_0(\mathbb{R}^n)$, para cada $f \in \mathcal{L}^1(\mathbb{R}^n)$. Teniendo esto en cuenta junto con la Proposición 4.2.1 y el Teorema 4.4.1, concluimos que la Transformada de Fourier

$$\hat{\cdot} : \mathcal{L}^1(\mathbb{R}^n) \rightarrow C_0(\mathbb{R}^n)$$

define un operador lineal y continuo.

4.6. Transformada de Fourier y Derivación

En esta sección se pretende analizar en detalle cómo se interrelacionan la Transformada de Fourier y la operación de derivación. Este análisis conlleva dos enfoques de estudio: uno que supone calcular la Transformada de Fourier de la derivada de una función, y relacionarla con la Transformada de Fourier de la función original; y otro que consiste en abordar la derivación de la función Transformada.

Comenzaremos por introducir un poco de notación. Para $\alpha = (\alpha_1, \alpha_2, \dots, \alpha_n) \in (\mathbb{N} \cup 0)^n$ e $y = (y_1, y_2, \dots, y_n) \in \mathbb{R}^n$ notaremos

$$|\alpha| = \sum_{k=1}^n \alpha_k \quad \text{e} \quad y^\alpha = \prod_{k=1}^n y_k^{\alpha_k} = y_1^{\alpha_1} y_2^{\alpha_2} \dots y_n^{\alpha_n}.$$

α se denominará multi-índice.

Si $f : \mathbb{R}^n \rightarrow \mathbb{C}$ es una función de clase C^∞ , escribimos

$$D_\alpha f = \frac{\partial^{|\alpha|} f}{\partial x_1^{\alpha_1} \partial x_2^{\alpha_2} \dots \partial x_n^{\alpha_n}}.$$

Para $|\alpha| = 0$, decimos que $D_\alpha f = f$.

Definición 4.6.1. Dado $k \in \{1, \dots, n\}$, denotaremos e_k al vector cuyas componentes son todas nulas salvo la k -ésima que es uno. Se tiene que, para $\alpha = e_k$,

$$D_\alpha f = \frac{\partial f}{\partial x_k}.$$

Denominaremos A_α al conjunto:

$$A_\alpha = \{\beta \in (\mathbb{N} \cup \{0\})^n : |\beta| \leq |\alpha|\}.$$

4.6.1. Transformada de la Derivada

Teorema 4.6.2. Dados $f \in \mathcal{L}^1(\mathbb{R}^n)$ y $k \in \{1, \dots, n\}$, si existe $\frac{\partial f}{\partial x_k}$ y además $\frac{\partial f}{\partial x_k} \in \mathcal{L}^1(\mathbb{R}^n)$, entonces

$$\widehat{\frac{\partial f}{\partial x_k}}(y) = (2\pi i y_k) \widehat{f}(y) \quad \forall y \in \mathbb{R}^n.$$

Equivalentemente, para $\alpha = e_k$,

$$\widehat{D_\alpha f}(y) = (2\pi i)^{|\alpha|} y^\alpha \widehat{f}(y) \quad \forall y \in \mathbb{R}^n.$$

Demostración.

Por hipótesis sabemos que dado $y \in \mathbb{R}^n$, la función $(x_1, \dots, x_n) \mapsto \frac{\partial f}{\partial x_k}(x_1, \dots, x_n) e^{-2\pi i \sum_{i=1}^n x_i y_i}$ es medible e integrable en $\mathbb{R}^n$:

$$\left| \frac{\partial f}{\partial x_k}(x) e^{-2\pi i \sum_{i=1}^n x_i y_i} \right| = \left| \frac{\partial f}{\partial x_k}(x) \right| < \infty.$$

La clave de la demostración radica en usar el Teorema de Fubini y posteriormente aplicar la fórmula de integración por partes en $\mathbb{R}$ a la función $x_k \mapsto \frac{\partial f}{\partial x_k}(x_1, \dots, x_n) e^{-2\pi i x_k y_k}$.

$$\begin{aligned} \widehat{\frac{\partial f}{\partial x_k}}(y) &= \int_{\mathbb{R}^n} \frac{\partial f}{\partial x_k}(x_1, \dots, x_n) e^{-2\pi i \sum_{i=1}^n x_i y_i} dx_1 \dots dx_{k-1} dx_k dx_{k+1} \dots dx_n \\ &= \int_{\mathbb{R}} \dots \left(\int_{\mathbb{R}} \frac{\partial f}{\partial x_k}(x_1, \dots, x_n) e^{-2\pi i x_k y_k} dx_k \right) e^{-2\pi i \sum_{i=1, i \neq k}^n x_i y_i} dx_1 \dots dx_{k-1} dx_{k+1} \dots dx_n \\ &= (2\pi i y_k) \int_{\mathbb{R}} \dots \int_{\mathbb{R}} f(x_1, \dots, x_n) e^{-2\pi i \sum_{i=1}^n x_i y_i} dx \\ &= (2\pi i y_k) \widehat{f}(y). \end{aligned} \quad \square$$

Teorema 4.6.3. Sea $\alpha \in (\mathbb{N} \cup \{0\})^n$ y supongamos que existe $D_\beta f \forall \beta \in A_\alpha$ de modo que

$$D_\beta f \in \mathcal{L}^1(\mathbb{R}^n) \quad \forall \beta \in A_\alpha.$$

Entonces

$$\widehat{D_\alpha f}(y) = (2\pi i)^{|\alpha|} y^\alpha \widehat{f}(y) \quad \forall y \in \mathbb{R}^n.$$

Demostración. Para un $\alpha \in (\mathbb{N} \cup \{0\})^n$ cualquiera, basta usar el Teorema 4.6.2 y realizar inducción sobre $|\alpha|$. $\square$

4.6.2. Derivada de la Transformada

Teorema 4.6.4. Sea $f \in \mathcal{L}^1(\mathbb{R}^n)$ tal que se cumple la condición

$$\int_{\mathbb{R}^n} |x_k f(x)| dx < \infty,$$

donde x_k denota la coordenada k -ésima de x . Entonces $\widehat{f}$ es derivable con respecto a y_k , y

$$\frac{\partial \widehat{f}}{\partial y_k}(y) = (-2\pi i x_k f(x))^\wedge(y) \quad \forall y \in \mathbb{R}^n.$$

Equivalentemente, para $\alpha = e_k$,

$$D_\alpha \widehat{f}(y) = (-2\pi i x^\alpha f(x))^\wedge(y) \quad \forall y \in \mathbb{R}^n.$$

Demostración. Sea la función $\gamma : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{C}$ definida por

$$\gamma(x, y) = f(x) e^{-2\pi i \langle x, y \rangle}.$$

Esta función es integrable con respecto la variable x y derivable con respecto a la variable y_k . Tenemos que:

$$\frac{\partial \gamma}{\partial y_k}(x, y) = \frac{\partial (f(x) e^{-2\pi i \langle x, y \rangle})}{\partial y_k} = -2\pi i x_k f(x) e^{-2\pi i \langle x, y \rangle}.$$

Además,

$$\left| \frac{\partial \gamma}{\partial y_k}(x, y) \right| = 2\pi |x_k f(x)| \quad \forall x, y \in \mathbb{R}^n.$$

Como por hipótesis la función $x_k f \in \mathcal{L}^1(\mathbb{R}^n)$, el Teorema de derivación de integrales dependientes de un parámetro asegura que $\widehat{f}$ es derivable con respecto a y_k , y también:

$$\begin{aligned} \frac{\partial \widehat{f}}{\partial y_k}(y) &= \int_{\mathbb{R}^n} \frac{\partial}{\partial y_k} (f(x) e^{-2\pi i \langle x, y \rangle}) dx \\ &= \int_{\mathbb{R}^n} (-2\pi i x_k) (f(x) e^{-2\pi i \langle x, y \rangle}) dx = (-2\pi i x_k f(x))^\wedge(y). \end{aligned} \quad \square$$

Corolario 4.6.5. Sea $f \in \mathcal{L}^1(\mathbb{R}^n)$. Supongamos que se cumple la condición

$$\int_{\mathbb{R}^n} |x_k^n f(x)| dx < \infty,$$

donde x_k denota la coordenada k -ésima de x . Entonces, $\widehat{f}$ es n veces derivable con respecto a

CAPÍTULO 4. TRANSFORMADA DE FOURIER EN $\mathcal{L}^1(\mathbb{R}^n)$

y_k , y se verifica que

$$\frac{\partial \hat{f}}{\partial y_k}(y) = ((-2\pi i x_k)^n f(x))^\wedge(y) \quad \forall y \in \mathbb{R}^n.$$

Equivalentemente, para $\alpha = n e_k$,

$$D_\alpha \hat{f}(y) = ((-2\pi i)^{|\alpha|} x^\alpha f(x))^\wedge(y) \quad \forall y \in \mathbb{R}^n.$$

Demostración. Para cada $j \in \{1, \dots, n\}$, tenemos que:

$$\int_{\mathbb{R}^n} |x_k^j f(x)| dx = \int_{|x_k| \leq 1} |x_k^j f(x)| dx + \int_{|x_k| > 1} |x_k^j f(x)| dx \leq \|f\|_1 + \int_{\mathbb{R}^n} |x_k^n f(x)| dx < \infty.$$

Luego se obtiene que $x_k^j f \in \mathcal{L}^1(\mathbb{R}^n)$. Realizando ahora una aplicación inductiva del teorema anterior, se demuestra el resultado buscado. $\square$

Teorema 4.6.6. Sea $f \in \mathcal{L}^1(\mathbb{R}^n)$ y $\alpha = (\alpha_1, \dots, \alpha_n) \in (\mathbb{N} \cup \{0\})^n$. Supongamos que se cumple la condición

$$\int_{\mathbb{R}^n} |x^\beta f(x)| dx < \infty \quad \forall \beta \in A_\alpha.$$

Entonces, se tiene que

$$D_\alpha \hat{f}(y) = ((-2\pi i)^{|\alpha|} x^\alpha f(x))^\wedge(y) \quad \forall y \in \mathbb{R}^n.$$

Demostración. Para un $\alpha \in (\mathbb{N} \cup \{0\})^n$ cualquiera, basta con usar el Teorema 4.6.4 y realizar inducción sobre $|\alpha|$. $\square$

4.7. Ejemplos

Definición 4.7.1. Definimos $G : \mathbb{R}^n \rightarrow \mathbb{R}$ función de $\mathcal{L}^1(\mathbb{R}^n)$ dada por

$$G(x) = e^{-\pi \|x\|^2} = e^{-\pi \sum_{k=1}^n x_k^2} \quad \forall x \in \mathbb{R}^n.$$

Observación 4.7.2. Dado $x = (x_1, x_2, \dots, x_n) \in \mathbb{R}^n$ se tiene que

$$G(x_1, x_2, \dots, x_n) = e^{-\pi x_1^2} \cdot e^{-\pi x_2^2} \dots e^{-\pi x_n^2} = G(x_1) \cdot G(x_2) \dots G(x_n) \quad (4.1)$$

Lema 4.7.3. G cumple

$$\int_{\mathbb{R}^n} G(x) dx = 1.$$

Demostración. Por un lado sabemos que

$$\hat{G}(0, 0, \dots, 0) = \int_{\mathbb{R}^n} G(x) dx.$$

Por otro, atendiendo a la expresión (4.7.2) y a la Proposición (4.2.6), se tiene que,

$$\widehat{G}(0, \dots, 0) = \widehat{G}(0) \cdots \widehat{G}(0),$$

luego podemos reducirnos al caso $n = 1$.

Para ello, haremos un cálculo previo que nos servirá en nuestro objetivo.

$$\int_{\mathbb{R}^2} e^{-\pi(x^2+y^2)} d(x, y) = \int_{-\pi}^{\pi} \int_0^{\infty} \rho e^{-\pi\rho^2} d\rho d\theta = \int_{-\pi}^{\pi} \frac{1}{2\pi} d\theta = 1.$$

Finalmente,

$$\int_{\mathbb{R}^2} e^{-\pi(x^2+y^2)} d(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} e^{-\pi x^2} e^{-\pi y^2} dy dx = \int_{-\infty}^{\infty} e^{-\pi x^2} dx \int_{-\infty}^{\infty} e^{-\pi y^2} dy = \left(\int_{-\infty}^{\infty} G(t) dt \right)^2.$$

De donde finalmente se deduce que para $G : \mathbb{R} \rightarrow \mathbb{R}$

$$\left(\int_{-\infty}^{\infty} G(t) dt \right)^2 = 1,$$

y por ende

$$\widehat{G}(0) = \left(\int_{-\infty}^{\infty} G(t) dt \right) = 1,$$

concluyendo lo que queríamos. □

Proposición 4.7.4. *G cumple la identidad*

$$\widehat{G} = G.$$

Demostración. Por un lado, atendiendo a la expresión (4.7.2) y a la Proposición (4.2.6), se tiene que, para $y = (y_1, y_2, \dots, y_n)$,

$$\widehat{G}(y_1, y_2, \dots, y_n) = \widehat{G}(y_1) \cdots \widehat{G}(y_n).$$

Luego podemos reducirnos al caso $n = 1$. Como

$$G'(x) = -2\pi x G(x) \quad \forall x \in \mathbb{R},$$

y además

$$\int_{\mathbb{R}} |G'(x)| dx = 2 \int_0^{\infty} 2\pi x e^{-\pi x^2} dx = 2,$$

el Teorema (4.6.4) garantiza que $\widehat{G}$ es derivable en $\mathbb{R}$. También,

$$\widehat{G}'(y) = (-2\pi i x G)^{\wedge}(y) = i \widehat{G}'(y) = -2\pi y \widehat{G}(y) \quad \forall y \in \mathbb{R}.$$

Luego $\widehat{G}$ es solución de la ecuación diferencial

$$f'(y) = -2\pi y f(y).$$

y, por tanto,

$$\widehat{G}(y) = ce^{-\pi y^2} = cG(y) \quad \forall y \in \mathbb{R},$$

y usando el Lema (4.7.3)

$$c = \widehat{G}(0) = \int_{\mathbb{R}} G(x) dx = 1.$$

Obtenemos finalmente $\widehat{G} = G$.

Volvemos ahora al caso general. Sea $y = (y_1, y_2, \dots, y_n)$, se verifica

$$\widehat{G}(y) = \prod_{j=1}^n \widehat{f}(y_j) = \prod_{j=1}^n f(y_j) = G(y). \quad \square$$

Observación 4.7.5. El proceso de filtrado de imágenes se emplea en el ámbito del procesamiento de imágenes con el objetivo de mejorar su calidad o de extraer información pertinente de las mismas. Consiste en aplicar diferentes operaciones matemáticas a los píxeles de una imagen con el fin de resaltar ciertas características, eliminar ruido o suavizar detalles no deseados. Este filtro recibe el nombre de núcleo o (kernel). Dentro del filtrado de imágenes, el núcleo gaussiano, que aparecerá con frecuencia en los siguientes ejemplos, es usado frecuentemente. Este núcleo nace de la discretización de la función gaussiana

$$G_{\sigma}(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \quad \sigma \in \mathbb{R}^+.$$

La distribución es extensamente conocida, y decae a medida que nos alejamos de su centro. Por tanto, como filtro discreto, le da más peso a los píxeles centrales y menos a los vecinos, “difuminando” los píxeles de la imagen. Por esto, es denominado un filtro de paso bajo o suavizado. El parámetro σ controla la cantidad de suavizado. Nótese que

$$G_{\sigma}(x) = \frac{1}{2\pi\sigma^2} H_{\frac{1}{\sqrt{\pi 2\sigma^2}}} G(x) \quad \forall x \in \mathbb{R}^2. \quad (4.2)$$

De (4.7.5) deducimos las siguientes características que simplifican el trabajo con la función gaussiana y contribuyen a su amplia utilización.

- La Transformada de Fourier de una función Gaussiana es otra función Gaussiana

$$\widehat{G}_{\sigma}(y) = \frac{1}{2\pi\sigma^2} (2\pi\sigma^2) \widehat{G}((2\pi\sigma^2)y) = G(\sqrt{2\pi\sigma^2}y) = G_{\sigma'}(y),$$

donde $\sigma' = \frac{1}{2\pi\sigma}$.

- Es separable, es decir se puede dividir en el producto de dos Gaussianas 1D.

Cuando nos refiramos a la derivada del núcleo gaussiano, será una discretización de la derivada de la función gaussiana salvo constante.

4.8. Teorema de Inversión

A continuación abordamos el Teorema de Inversión que nos permitirá recuperar la función a partir de su Transformada de Fourier bajo ciertas hipótesis.

Teorema 4.8.1 (de Inversión). *Sea $f \in \mathcal{L}^1(\mathbb{R}^n)$ y supongamos que $\widehat{f} \in \mathcal{L}^1(\mathbb{R}^n)$. Entonces*

$$f(x) = \int_{\mathbb{R}^n} \widehat{f}(y) e^{2\pi i \langle x, y \rangle} dy \quad \text{para casi todo } x \in \mathbb{R}^n. \quad (4.3)$$

Demostración. Sea la función G definida en la Definición (4.7.1). Dado $x \in \mathbb{R}^n$, probaremos primero que

$$\lim_{m \rightarrow \infty} \int_{\mathbb{R}^n} \widehat{f}(y) e^{2\pi i \langle x, y \rangle} G\left(\frac{1}{m}y\right) dy = \int_{\mathbb{R}^n} \widehat{f}(y) e^{2\pi i \langle x, y \rangle} dy. \quad (4.4)$$

Para ello, observamos los siguientes puntos:

- $\lim_{m \rightarrow \infty} \widehat{f}(y) e^{2\pi i \langle x, y \rangle} G\left(\frac{1}{m}y\right) = \widehat{f}(y) e^{2\pi i \langle x, y \rangle},$
- $|\widehat{f}(y) e^{2\pi i \langle x, y \rangle} G\left(\frac{1}{m}y\right)| \leq |\widehat{f}(y)| \quad \forall y \in \mathbb{R}^n, \forall m \in \mathbb{N},$
- $|\widehat{f}|$ es por hipótesis integrable en $\mathbb{R}^n$.

Aplicando el Teorema de la convergencia dominada se concluye (4.8).

Por otro lado, usando el Lema 4.7.3 y el Teorema 4.3.1, se tiene que

$$\begin{aligned} & \int_{\mathbb{R}^n} \widehat{f}(y) e^{2\pi i \langle x, y \rangle} G\left(\frac{1}{m}y\right) dy - f(x) = \\ &= \int_{\mathbb{R}^n} \widehat{f}(y) e^{2\pi i \langle x, y \rangle} G\left(\frac{1}{m}y\right) dy - \int_{\mathbb{R}^n} G(y) f(x) dy \\ &= \int_{\mathbb{R}^n} \widehat{\tau_{-x}f}(y) G\left(\frac{1}{m}y\right) dy - \int_{\mathbb{R}^n} G(y) f(x) dy \\ &= \int_{\mathbb{R}^n} \tau_{-x}f(y) \widehat{G}\left(\frac{1}{m}y\right) dy - \int_{\mathbb{R}^n} G(y) f(x) dy \\ &= \int_{\mathbb{R}^n} \tau_{-x}f(y) m^n G(my) dy - \int_{\mathbb{R}^n} G(y) dy \\ &= \int_{\mathbb{R}^n} \left[f\left(\frac{y}{m} + x\right) - f(x) \right] G(y) dy. \end{aligned}$$

Luego

$$\left| \int_{\mathbb{R}^n} \widehat{f}(y) e^{2\pi i \langle x, y \rangle} G\left(\frac{1}{m}y\right) dy - f(x) \right| \leq \int_{\mathbb{R}^n} \left| f\left(\frac{y}{m} + x\right) - f(x) \right| G(y) dy.$$

Integrando con respecto a $x \in \mathbb{R}^n$ se tiene

$$\begin{aligned} \int_{\mathbb{R}^n} \left| \int_{\mathbb{R}^n} \widehat{f}(y) e^{2\pi i \langle x, y \rangle} G\left(\frac{1}{m}y\right) dy - f(x) \right| dx &\leq \int_{\mathbb{R}^n} \int_{\mathbb{R}^n} \left| f\left(\frac{y}{m} + x\right) - f(x) \right| G(y) dx dy \\ &= \int_{\mathbb{R}^n} w_1 f\left(\frac{1}{m}y\right) G(y) dy. \end{aligned}$$

Recurriendo de nuevo al Teorema de la convergencia dominada, probaremos que

$$\lim_{m \rightarrow \infty} \int_{\mathbb{R}^n} w_1 f \left(\frac{1}{m} y \right) G(y) dy = 0.$$

Observamos los siguientes puntos:

- $\lim_{m \rightarrow \infty} w_1 f \left(\frac{1}{m} y \right) G(y) = 0$,
- $w_1 f \left(\frac{1}{m} y \right) G(y) \leq 2 \|f\|_1 G(y) \quad \forall y \in \mathbb{R}^n, \forall m \in \mathbb{N}$,
- La función $\|f\|_1 G$ es integrable en $\mathbb{R}^n$.

Por lo que se verifican todas las hipótesis para poder aplicar de nuevo el Teorema de la convergencia dominada para obtener que

$$\lim_{m \rightarrow \infty} \int_{\mathbb{R}^n} \left| \int_{\mathbb{R}^n} \widehat{f}(y) e^{2\pi i \langle x, y \rangle} G \left(\frac{1}{m} y \right) dy - f(x) \right| dx = 0.$$

Esto, junto con (4.8), nos permite concluir que

$$\int_{\mathbb{R}^n} \widehat{f}(y) e^{2\pi i \langle x, y \rangle} dy = f(x) \quad \text{c.p.d. en } \mathbb{R}^n. \quad \square$$

Corolario 4.8.2. *La fórmula que aparece en (4.8.1) se cumple para cualquier punto en el que f es continua.*

Demostración. Definimos la función $F : \mathbb{R}^n \mapsto \mathbb{C}$ por

$$F(x) = \int_{\mathbb{R}^n} \widehat{f}(y) e^{2\pi i \langle x, y \rangle} dy \quad \forall x \in \mathbb{R}^n.$$

Dado $a \in \mathbb{R}^n$, supongamos que f es continua en a y razonemos por reducción al absurdo. Si $F(a) \neq f(a)$, entonces $\exists \delta \in \mathbb{R}^+$ tal que $F(x) \neq f(x) \quad \forall x \in]a - \delta, a + \delta[$, por ser F y f funciones continuas en el punto a . Como consecuencia, existiría un conjunto de medida no nula $]a - \delta, a + \delta[$ en el que no se verifica (4.8.1) contrariando lo obtenido en el Teorema 4.8.1. $\square$

Es interesante puntualizar que la condición de que la Transformada de Fourier sea integrable, es bastante restrictiva y en general no será cierta.

Sin embargo en el ámbito discreto, como es de esperar, podremos recuperar cualquier señal a partir de la Transformada usando la Inversa de la Transformada de Fourier Discreta (IDFT).

4.9. Teorema de Unicidad

El Teorema de Unicidad sugiere que la Transformada de Fourier $\widehat{f}$ puede interpretarse como el código genético de la función f . En efecto, si dos funciones integrables en $\mathbb{R}^n$ tienen la misma Transformada de Fourier, esto implicará que son idénticas salvo en un conjunto de medida nula. Por tanto, cabe pensar que la Transformada de Fourier, en cierto sentido, sintetiza la función original.

Teorema 4.9.1. (*Teorema de Unicidad*). Sean $f, g \in \mathcal{L}^1(\mathbb{R}^n)$ y supongamos que $\widehat{f}(y) = \widehat{g}(y)$ para todo $y \in \mathbb{R}^n$. Entonces $f = g$ c.p.d. en $\mathbb{R}^n$.

Demostración. La prueba se deduce del Teorema de Inversión.

Si $f, g \in \mathcal{L}^1(\mathbb{R}^n)$, entonces $f - g \in \mathcal{L}^1(\mathbb{R}^n)$ con $\widehat{f}(y) - \widehat{g}(y) = \widehat{f - g}(y) = 0 \quad \forall y \in \mathbb{R}^n$. Por tanto, el Teorema de Inversión es aplicable a la función $f - g$ y deducimos que

$$f(x) - g(x) = \int_{\mathbb{R}^n} (\widehat{f - g})(y) e^{2\pi i \langle x, y \rangle} dy = 0. \quad \text{c.p.d. en } \mathbb{R}^n,$$

de donde deducimos que $f = g$ c.p.d. en $\mathbb{R}^n$. $\square$

4.10. Clase de Schwartz

Tal como se ha señalado previamente, existen referencias, tales como [22] donde se desarrolla la Transformada de Fourier en una clase más restringida que $\mathcal{L}^1(\mathbb{R}^n)$, el espacio de funciones de Schwartz $\mathcal{S}(\mathbb{R}^n)$, que resulta ser un entorno particularmente adecuado para esta. Es por ello por lo que estudiaremos algunas características de este espacio, así como la razón de que el comportamiento de la Transformada sea especialmente bueno en él.

Definición 4.10.1. El espacio de Schwartz, denotado por $\mathcal{S}(\mathbb{R}^n)$, se define como el conjunto de todas las funciones $f : \mathbb{R}^n \rightarrow \mathbb{C}$ que son infinitamente diferenciables, de modo que para cada par de multi-índices α y $\beta \in (\mathbb{N} \cup 0)^n$, se tiene que

$$\rho_{\beta, \alpha}(f) := \sup_{x \in \mathbb{R}^n} |x^\beta D_\alpha f(x)| < \infty.$$

Observación 4.10.2. Las funciones de $\mathcal{S}(\mathbb{R}^n)$ son funciones infinitamente diferenciables cuyas sucesivas derivadas decrecen más rápido en infinito que cualquier polinomio.

Es posible proporcionar otra definición utilizando un único multi-índice.

Definición 4.10.3. Sea $f \in C^\infty(\mathbb{R}^n)$. Se dice que f pertenece al espacio de Schwartz si es indefinidamente derivable y para cada $m \in \mathbb{N}$ y cada multi-índice $\alpha \in (\mathbb{N} \cup \{0\})^n$ se cumple que

$$\rho_{m, \alpha}(f) := \sup_{x \in \mathbb{R}^n} (1 + \|x\|^2)^m |D_\alpha f(x)| < \infty.$$

Se puede demostrar con relativa facilidad que ambas definiciones son equivalentes. Se ha usado a propósito la misma notación para los dos supremos.

Proposición 4.10.4. Si $f \in \mathcal{S}(\mathbb{R}^n)$ entonces $f \in \mathcal{L}^1(\mathbb{R}^n)$.

Demostración. Tomamos $\alpha = 0$. Como $f \in \mathcal{S}(\mathbb{R}^n)$, se tiene que

$$|f(x)|(1 + \|x\|^2)^n \leq \rho_{n, \alpha},$$

Usando coordenadas polares se tiene que

$$\int_{\mathbb{R}^n} |f(x)| dx = \int_{\mathbb{R}^n} \frac{\rho_{n, \alpha}}{(1 + \|x\|^2)^n} dx = \rho_{n, \alpha} w_{n-1} \int_0^\infty \frac{r^{n-1}}{(1 + r^2)^n} dr < \infty,$$

donde w_{n-1} denota el área de $\mathbb{S}^{n-1}$. □

Proposición 4.10.5. Si $f, g \in \mathcal{S}(\mathbb{R}^n)$, entonces se tiene que

- $f + g \in \mathcal{S}(\mathbb{R}^n)$,
- $\gamma f \in \mathcal{S}(\mathbb{R}^n) \quad \forall \gamma \in \mathbb{C}$.

Demostración. Probaremos a continuación los dos apartados.

- Veamos que $f + g \in \mathcal{S}(\mathbb{R}^n)$. Para cada par de multi-índices $\alpha, \beta \in (\mathbb{N} \cup 0)^n$, se tiene que

$$D_\alpha(f + g) = D_\alpha(f) + D_\alpha(g).$$

Luego

$$\begin{aligned} |x^\beta D_\alpha(f + g)| &= |x^\beta| |D_\alpha(f) + D_\alpha(g)| \\ &\leq |x^\beta| |D_\alpha(f)| + |x^\beta| |D_\alpha(g)| \\ &\leq \rho_{\beta, \alpha}(f) + \rho_{\beta, \alpha}(g). \end{aligned}$$

Y, por tanto,

$$\rho_{\beta, \alpha}(f + g) \leq \rho_{\beta, \alpha}(f) + \rho_{\beta, \alpha}(g) < \infty.$$

Concluimos que $f + g \in \mathcal{S}(\mathbb{R}^n)$.

- Sea $\gamma \in \mathbb{C}$. Veamos que $\gamma f \in \mathcal{S}(\mathbb{R}^n)$. Para cada par de multi-índices $\alpha, \beta \in (\mathbb{N} \cup 0)^n$, se tiene que

$$D_\alpha(\gamma f) = \gamma D_\alpha(f).$$

Luego

$$|x^\beta D_\alpha(\gamma f)| = |x^\beta \gamma D_\alpha(f)| = |\gamma| |x^\beta D_\alpha(f)| \leq |\gamma| \rho_{\beta, \alpha}(f).$$

Y, por tanto,

$$\rho_{\beta, \alpha}(\gamma f) \leq |\gamma| \rho_{\beta, \alpha}(f) < \infty,$$

obteniendo que $\gamma f \in \mathcal{S}(\mathbb{R}^n)$. □

Observación 4.10.6. Las Proposiciones 4.10.4 y 4.10.5 muestran que el espacio $\mathcal{S}(\mathbb{R}^n)$ es un subespacio vectorial de $\mathcal{L}^1(\mathbb{R}^n)$. Consecuentemente, la teoría desarrollada para $\mathcal{L}^1(\mathbb{R}^n)$ se aplica directamente a este espacio.

Proposición 4.10.7. Si $f \in \mathcal{S}(\mathbb{R}^n)$, entonces $\widehat{f} \in \mathcal{S}(\mathbb{R}^n)$.

Demostración. Probaremos primero que $\widehat{f}$ es infinitamente derivable. Como puede intuir el lector, pretendemos usar el Teorema 4.6.4. Para ello tenemos que demostrar que si $f \in \mathcal{S}(\mathbb{R}^n)$, se verifican las hipótesis de este teorema.

Sea $\alpha = (\alpha_1, \dots, \alpha_n)$, atendiendo a la equivalencia de las dos definiciones de $\mathcal{S}(\mathbb{R}^n)$ proporcionadas se tiene que

$$\int_{\mathbb{R}^n} |x^\alpha f(x)| dx < \infty.$$

Por tanto, se tiene que $\widehat{f}$ es infinitamente derivable y además

$$D_\alpha \widehat{f}(y) = ((-2\pi i)^{|\alpha|} x^\alpha f(x))^\wedge(y) \quad \forall y \in \mathbb{R}^n.$$

Definimos la función

$$g(x) = (-2\pi i)^{|\alpha|} x^\alpha f(x) \quad \forall x \in \mathbb{R}^n.$$

Dado $\beta \in (\mathbb{N} \cup 0)^n$,

$$y^\beta D_\alpha \widehat{f}(y) = y^\beta \widehat{g(x)}(y) = \widehat{D_\beta g(x)}(y) \frac{1}{(2\pi i)^{|\beta|}},$$

de donde se tiene que $y \mapsto y^\beta D_\alpha \widehat{f}$ está acotada $\forall \beta \in (\mathbb{N} \cup 0)^n$, ya que es la Transformada de Fourier de una función integrable en $\mathbb{R}^n$. Como consecuencia deducimos que $\widehat{f} \in \mathcal{S}(\mathbb{R}^n)$. $\square$

Observación 4.10.8. Trabajando el concepto de la Transformada de Fourier en $\mathcal{S}(\mathbb{R}^n)$, tenemos garantizado de antemano el buen planteamiento de su construcción inversa, ya que para poder aplicar el Teorema de Inversión a una función f necesitábamos que $\widehat{f}$ fuera integrable, una condición que gracias a la proposición anterior se cumple para $f \in \mathcal{S}(\mathbb{R}^n)$.

Proposición 4.10.9. *El operador*

$$\widehat{\cdot}: \mathcal{S}^1(\mathbb{R}^n) \rightarrow \mathcal{S}^1(\mathbb{R}^n)$$

define una biyección.

Demostración. Sabemos por la Proposición 4.10.7 que $\widehat{f} \in \mathcal{S}(\mathbb{R}^n)$ para cada $f \in \mathcal{S}(\mathbb{R}^n)$. Además,

- Es inyectivo como consecuencia del Teorema de Unicidad. Esto es: si $\widehat{f} = 0$, se tiene que $f = 0$ c.p.d.
- Es sobreyectivo por el Teorema de Inversión. Dada una función $g \in \mathcal{S}(\mathbb{R}^n)$, definimos la función $f = \widetilde{g}$, de tal manera que por el Teorema de Inversión se tiene justamente que $\widehat{f} = g$.

$\square$

La familia numerable de seminormas

$$\sup_{|\alpha| \leq N} \sup_{x \in \mathbb{R}^n} (1 + \|x\|_2^2)^k |D_\alpha f(x)| \quad (f \in \mathcal{S}(\mathbb{R}^d), k, N \in \mathbb{N}),$$

da lugar a una topología en $\mathcal{S}(\mathbb{R}^n)$, localmente convexa y metrizable, generada por intersecciones finitas de bolas definidas con respecto a tales seminormas. El dual topológico de $\mathcal{S}(\mathbb{R}^n)$, denotado $\mathcal{S}'(\mathbb{R}^n)$, está formado por todos los funcionales lineales y continuos definidos sobre $\mathcal{S}(\mathbb{R}^n)$, y sus elementos reciben el nombre de distribuciones temperadas. En este espacio se puede dar una definición de la Transformada de Fourier y probar que la Transformada de Fourier de una distribución temperada es otra distribución temperada. También se puede probar que la Transformada de Fourier es un homeomorfismo lineal de $\mathcal{S}(\mathbb{R}^n)$ en $\mathcal{S}(\mathbb{R}^n)$, y de $\mathcal{S}'(\mathbb{R}^n)$ en $\mathcal{S}'(\mathbb{R}^n)$. Esta aproximación, aunque escapa del alcance del presente análisis, se presenta como una línea prometedora para trabajo futuro.

Capítulo 5.

Convolución

5.1. Definición

En el presente capítulo, exploramos la operación de convolución, una componente fundamental en la teoría del Análisis de Fourier. En la segunda parte, estudiaremos aplicaciones en el ámbito computacional, donde por ejemplo el filtrado de imágenes es gobernado por esta operación, y el uso del Teorema de Convolución que será sin lugar a duda el eje central del trabajo.

Definición 5.1.1. Sean $f, g \in \mathcal{L}^0(\mathbb{R}^n)$. Dado $x \in \mathbb{R}^n$, decimos que la *convolución* está definida en $x \in \mathbb{R}^n$ si se cumple la condición

$$\int_{\mathbb{R}^n} |f(x-y)g(y)| dy < \infty.$$

En esa situación, se establece la definición

$$(f * g)(x) = \int_{\mathbb{R}^n} f(x-y)g(y) dy < \infty.$$

Observación 5.1.2. El concepto de convolución tiene perfecto sentido en funciones definidas c.p.d. en $\mathbb{R}^n$, ya que el valor de la convolución (cuando proceda) de dos funciones en un punto es completamente independiente de las posibles alteraciones de estas en conjuntos de medida nula.

A continuación, probaremos que si $f, g \in \mathcal{L}^1(\mathbb{R}^n)$, entonces la convolución está definida para todo punto en $\mathbb{R}^n$.

Lema 5.1.3. Sean $f, g \in \mathcal{L}^0(\mathbb{R}^n)$. La función $\gamma : \mathbb{R}^{n+n} \rightarrow \mathbb{C}$ con

$$\gamma(x, y) = f(x-y)g(y) \quad \forall x, y \in \mathbb{R}^n.$$

es medible.

Demostración. Se definen las funciones $f_1, g_2 : \mathbb{R}^{n+n} \rightarrow \mathbb{C}$ como

$$f_1(x, y) = f(x) \quad g_2(x, y) = g(y) \quad \forall x, y \in \mathbb{R}^n.$$

CAPÍTULO 5. CONVOLUCIÓN

Por otro lado, se define la biyección lineal $\phi : \mathbb{R}^{n+n} \rightarrow \mathbb{R}^{n+n}$ por:

$$\phi(x, y) = (x - y, y) \quad x, y \in \mathbb{R}^n.$$

Se observa que $\gamma = (f_1 g_2) \circ \phi$. Luego la medibilidad de γ se reduce a probar que las funciones f_1 y g_2 lo son. En efecto, dado $A \in \mathcal{B}(\mathbb{C})$, se sigue que

$$(f_1)^{-1}(A) = (f)^{-1}(A) \times \mathbb{R}^n, \quad (5.1)$$

$$(g_2)^{-1}(A) = \mathbb{R}^n \times (g)^{-1}(A). \quad (5.2)$$

Usando que f, g son funciones medibles, se tiene que (5.1) y (5.2) son conjuntos medibles. Llegamos así a la medibilidad de f_1, g_2 y por tanto también a la de γ . $\square$

Teorema 5.1.4. Sean $f, g \in \mathcal{L}^1(\mathbb{R}^n)$. Entonces se tiene que

1. $f * g$ está definida c.p.d en $\mathbb{R}^n$,
2. $f * g \in \mathcal{L}^1(\mathbb{R}^n)$ y además se cumple la desigualdad

$$\|f * g\|_1 \leq \|f\|_1 \|g\|_1.$$

Demostración. Consideramos la función γ definida en el Lema 5.1.3, medible. A continuación, probamos que

$$\int_{\mathbb{R}^{n+n}} |\gamma(x, y)| d(x, y) = \|f\|_1 \|g\|_1 < \infty.$$

Para ello observamos que

$$\int_{\mathbb{R}^n} \int_{\mathbb{R}^n} |\gamma(x, y)| dx dy = \int_{\mathbb{R}^n} |g(y)| \int_{\mathbb{R}^n} |f(x - y)| dx dy = \int_{\mathbb{R}^n} |g(y)| \|f\|_1 dy = \|f\|_1 \|g\|_1.$$

Aplicando el Teorema de Tonelli, se tiene que

$$\int_{\mathbb{R}^{n+n}} |\gamma(x, y)| d(x, y) = \int_{\mathbb{R}^n} \int_{\mathbb{R}^n} |\gamma(x, y)| dx dy = \|f\|_1 \|g\|_1 < \infty.$$

Por el Teorema de Fubini, se sigue ahora que:

- Para casi todo $x \in \mathbb{R}^n$, la función $y \mapsto \gamma(x, y)$ es integrable en $\mathbb{R}^n$.
- La función $x \mapsto \int_{\mathbb{R}^n} \gamma(x, y) dy$ definida c.p.d en $\mathbb{R}^n$ es integrable en $\mathbb{R}^n$.

Por lo que $f * g$ es integrable en $\mathbb{R}^n$, y se tiene que

$$\begin{aligned} \|f * g\|_1 &= \int_{\mathbb{R}^n} |(f * g)(x)| dx \leq \int_{\mathbb{R}^n} \int_{\mathbb{R}^n} |\gamma(x, y)| dy dx \\ &= \int_{\mathbb{R}^n} \int_{\mathbb{R}^n} |\gamma(x, y)| dx dy = \|f\|_1 \|g\|_1. \end{aligned} \quad \square$$

5.2. Teorema de Convolución

A continuación, probaremos el teorema que servirá de eje central para alcanzar el objetivo propuesto en el presente trabajo. Podemos afirmar, en términos coloquiales, que la Transformada convierte la convolución en el producto. En la segunda parte de la memoria, nos centraremos en aprovechar esta conversión para expresar la convolución entre imágenes como productos puntuales, reduciendo la complejidad computacional considerablemente.

Teorema 5.2.1. (*Teorema de Convolución*). Sean $f, g \in \mathcal{L}^1(\mathbb{R}^n)$. Entonces se tiene que

$$\widehat{(f * g)}(y) = \widehat{f}(y)\widehat{g}(y) \quad \forall y \in \mathbb{R}^n.$$

Demostración. Sea γ la función introducida en (5.1.3) y $z \in \mathbb{R}^n$. La función $(x, y) \mapsto \gamma(x, y)e^{-2\pi i \langle x, z \rangle}$ es

- medible en $\mathbb{R}^n \times \mathbb{R}^n$,
- integrable en $\mathbb{R}^n \times \mathbb{R}^n$ ya que:

$$\int_{\mathbb{R}^{n+n}} |\gamma(x, y)e^{-2\pi i \langle x, z \rangle}| d(x, y) = \int_{\mathbb{R}^{n+n}} |\gamma(x, y)| d(x, y) < \infty.$$

Aplicamos entonces el Teorema de Fubini y obtenemos:

$$\begin{aligned} \widehat{(f * g)}(z) &= \int_{\mathbb{R}^n} (f * g)(x) e^{-2\pi i \langle x, z \rangle} dx \\ &= \int_{\mathbb{R}^n} \left(\int_{\mathbb{R}^n} (f(x - y)g(y) dy e^{-2\pi i \langle x, z \rangle} \right) dx \\ &= \int_{\mathbb{R}^n} \int_{\mathbb{R}^n} (f(x - y)g(y) e^{-2\pi i \langle x, z \rangle} e^{-2\pi i \langle y, z \rangle} e^{2\pi i \langle y, z \rangle} dy dx \\ &= \int_{\mathbb{R}^n} \left(\int_{\mathbb{R}^n} (f(x - y) e^{-2\pi i \langle (x - y), z \rangle} dx g(y) e^{-2\pi i \langle y, z \rangle} \right) dy \\ &= \int_{\mathbb{R}^n} f(x) e^{-2\pi i \langle x, z \rangle} dx \int_{\mathbb{R}^n} g(y) e^{-2\pi i \langle y, z \rangle} dy \\ &= \widehat{f}(z)\widehat{g}(z). \end{aligned}$$

□

El Teorema de Convolución permite en el ámbito discreto interpretar el filtrado de imágenes en términos de frecuencias, ya que este se puede ver como el producto puntual de las representaciones de dichas imágenes en el dominio de la frecuencia. Esto proporciona una mayor explicabilidad acerca del resultado obtenido tras el filtrado.

Por ejemplo, imagínese que tratamos de eliminar el ruido de una imagen. Esto en la práctica se puede hacer realizando la convolución de la imagen con un núcleo gaussiano G_σ . Pero si queremos pensar en términos de la frecuencia podemos realizar la operación de convolución usando el Teorema 5.2.1. Al calcular la Transformada de Fourier Discreta de la imagen, obtenemos su representación en el dominio de la frecuencia. Por otro lado, al calcular la Transformada de Fourier del núcleo gaussiano, obtenemos otro núcleo gaussiano $G_{\sigma'}$, que al multiplicar punto a punto con la Transformada de la imagen, elimina aquellas frecuencias altas (las multiplica por cero), al ser un filtro de paso bajo. Por lo que al destransformar el resultado, obtenemos la imagen sin ruido.

Este teorema no solo es relevante en el ámbito computacional. Sin ir más lejos, tiene aplicaciones directas dentro del Análisis de Fourier. Este puede resultar especialmente útil en la resolución de ecuaciones que involucran la convolución, como ilustraremos en el siguiente ejemplo.

Ejemplo 5.2.2. Supongamos $f \in \mathcal{L}^1(\mathbb{R}^n)$ satisface la ecuación

$$2024f * f * f * f + f * f * f - 2f * f + f = 0 \text{ c.p.d. en } \mathbb{R}^n. \quad (5.3)$$

Entonces se tiene que $f = 0$ c.p.d. en $\mathbb{R}^n$.

Resolución. Suponemos que existe $f \in \mathcal{L}^1(\mathbb{R}^n)$ tal que satisface la ecuación (5.2). Por el Teorema de Convolución sabemos que

$$0 = (2024f * f * f * f + f * f * f - 2f * f + f)^\wedge = 2024\hat{f}(y)^4 + \hat{f}(y)^3 - 2\hat{f}(y)^2 + \hat{f}(y) \quad \forall y \in \mathbb{R}^n.$$

Tenemos así que $\hat{f}(\mathbb{R}^n) \subset \mathcal{Z}$, donde

$$\mathcal{Z} = \{0, w_1, w_2, w_3\}, \quad w_1, w_2, w_3 \in \mathbb{C}.$$

Como $\hat{f}$ es una función continua, y $\mathbb{R}^n$ es conexo, se concluye que $\hat{f}(\mathbb{R}^n)$ es un subconjunto conexo de $\mathcal{Z}$. Como los únicos subconjuntos conexos de $\mathcal{Z}$ son los conjuntos que contienen un único punto, concluimos que $\hat{f}$ es constante, y se cumple una de las siguientes opciones:

- $\hat{f} = 0$.
- $\hat{f} = w$, $w \in \{w_1, w_2, w_3\}$.

No obstante, de acuerdo con el Lema de Riemann-Lebesgue, necesariamente $\lim_{||y|| \rightarrow \infty} \hat{f}(y) = 0$.

Esto nos lleva a deducir que la única alternativa es que $\hat{f}(y) = 0 \forall y \in \mathbb{R}^n$. El Teorema de Unicidad asegura por tanto que $f = 0$ c.p.d. Es interesante observar que si

$$2024f * f * f * f + f * f * f - 2f * f + f = 0 \text{ p.d. en } \mathbb{R}^n, \quad (5.4)$$

se concluye que $f = 0$ p.d. en $\mathbb{R}^n$. En efecto, de lo razonado anteriormente, se tendría que $f = 0$ c.p.d. Lo cual implicaría que los términos donde aparece la operación de convolución están definidos en todo punto y son idénticamente nulos. Por lo que se tendría

$$f(x) = -(2024f * f * f * f)(x) - (f * f * f)(x) + (2f * f)(x) = 0 \text{ p.d. en } \mathbb{R}^n.$$

Observación 5.2.3. En la mayoría de textos que abordan la convolución para su aplicación en el ámbito computacional encontramos tras este teorema la fórmula $\widehat{fg} = \hat{f} * \hat{g}$. De manera que se sugiere que la Transformada convierte la operación de convolución en el producto y viceversa. Así, la convolución en el dominio del tiempo se transforma en el producto en el dominio de la frecuencia, y la multiplicación en el dominio del tiempo corresponde a la convolución en el dominio de la frecuencia. Puede por tanto extrañar que esta no aparezca en esta sección. Sin embargo, es importante destacar que si f y g pertenecen al espacio de las funciones integrables $\mathcal{L}^1(\mathbb{R}^n)$, el producto puntual fg no necesariamente pertenece a $\mathcal{L}^1(\mathbb{R}^n)$. Y por tanto, ahora mismo no podemos abordar esta cuestión. No obstante, esta ecuación se retomará en un contexto diferente más adelante. Una vez que cambiemos el marco de trabajo adecuadamente esta tomará pleno sentido.

5.3. Propiedades de la Convolución

Terminaremos el capítulo probando algunas propiedades de la convolución. La demostración de estas proposiciones será inmediata empleando el Teorema de Convolución, lo que subraya nuevamente su importancia.

Observación 5.3.1. Las igualdades que aparecen en la siguiente proposición son casi por doquier en $\mathbb{R}^n$.

Proposición 5.3.2. Sea $f, g, h \in \mathcal{L}^1(\mathbb{R}^n)$ y sea $\alpha \in \mathbb{C}$. Entonces se tiene que

- $f * g = g * f$,
- $(f + g) * h = f * h + g * h$,
- $(\alpha f) * h = \alpha(f * h)$,
- $(f * g) * h = f * (g * h)$.

Demostración. Por un lado el Teorema 5.2.1 nos permite trasladar las operaciones conmutativa y distributiva de la convolución a la de los números complejos vía la Transformada de Fourier. Seguidamente el Teorema de Unicidad nos proporciona las igualdades casi por doquier correspondientes. $\square$

Proporcionaremos al lector una breve motivación acerca del siguiente resultado que puede pasar inadvertida. La Proposición 5.3.3 (que llevaremos más adelante al ámbito discreto) es crucial en el contexto de las redes convolucionales. Esta es conocida como la *equivariancia de la traslación*, la cual expresa que si trasladamos una imagen y realizamos la convolución obtenemos el mismo resultado que si trasladamos la imagen convolucionada. Su importancia reside en que un objeto de una imagen no tiene por qué estar fijo para que pueda ser detectada por una *CNN* (Red neuronal convolucional); el sistema funciona de la misma manera en diferentes ubicaciones, pero su respuesta cambia a medida que varía la ubicación del objetivo. Por tanto, la red no necesita aprender diferentes representaciones para objetos ubicados en diferentes partes de la imagen.

Proposición 5.3.3. Sea $f \in \mathcal{L}^1(\mathbb{R}^n)$ y sea $t \in \mathbb{R}^n$. Entonces

$$\tau_t(f * g) = \tau_t(f) * g = f * \tau_t(g). \quad (5.5)$$

Demostración. Usando la Proposición 4.2.4, obtenemos

$$\widehat{\tau_t(f * g)}(y) = e^{-2\pi i \langle t, y \rangle} \widehat{(f * g)}(y) = e^{-2\pi i \langle t, y \rangle} \widehat{f}(y) \widehat{g}(y) = \widehat{\tau_t f}(y) \widehat{g}(y) = (\tau_t(f) * g)^\wedge(y).$$

Posteriormente, el Teorema de Unicidad asegura la primera igualdad de (5.3.3). Para la segunda basta con intercambiar f y g . $\square$

Proposición 5.3.4. Sea $f, g \in \mathcal{L}^1(\mathbb{R}^n)$. Entonces

- $\overline{f * g} = \overline{f} * \overline{g}$,
- $\widetilde{f * g} = \widetilde{f} * \widetilde{g}$,
- $\overline{\widetilde{(f * g)}(y)} = \overline{\widetilde{f}} \overline{\widetilde{g}}.$

CAPÍTULO 5. CONVOLUCIÓN

Demostración. La prueba es inmediata usando el Teorema 5.2.1 y el Teorema de Unicidad. $\square$

Proposición 5.3.5. Sean $f, g \in \mathcal{L}^1(\mathbb{R}^n)$, y $a \in \mathbb{R}^+$. Entonces

$$H_a(f * g) = a^{-n}(H_a f) * (H_a g). \quad (5.6)$$

Demostración.

$$\widehat{H_a(f * g)} = \frac{1}{a^n} \widehat{(f * g)} \left(\frac{1}{a} \right) = \frac{1}{a^n} \widehat{f} \left(\frac{1}{a} \right) \widehat{g} \left(\frac{1}{a} \right).$$

$$(a^n(H_a f) * (H_a g))^\wedge = a^n \widehat{H_a f} \widehat{H_a g} = a^n \frac{1}{a^n} \widehat{f} \left(\frac{1}{a} \right) \frac{1}{a^n} \widehat{g} \left(\frac{1}{a} \right) = \widehat{f} \left(\frac{1}{a} \right) \frac{1}{a^n} \widehat{g} \left(\frac{1}{a} \right).$$

Usando el Teorema de Unicidad, concluimos (5.3.5). $\square$

Definición 5.3.6. Sean $f, g \in \mathcal{L}^1(\mathbb{R}^n)$. Denominaremos correlación de f, g a la operación $f \star g$ dada por

$$(f \star g)(x) = \int_{\mathbb{R}^n} \overline{f(y)} g(x + y) dy \quad \text{para los } x \in \mathbb{R}^n \text{ en los que tenga sentido.}$$

Proposición 5.3.7. Sea $f, g \in \mathcal{L}^1(\mathbb{R}^n)$. Entonces $(f \star g)$ está definida c.p.d en $\mathbb{R}^n$ y además se tiene que

$$(f \star g)(x) = (\widetilde{f} * g)(x).$$

Demostración. Por el Teorema 5.1.4, la convolución está definida c.p.d en $\mathbb{R}^n$. Tomamos un punto $x \in \mathbb{R}^n$ para el que está definida. Entonces se sigue que

$$(\widetilde{f} * g)(x) = \int_{\mathbb{R}^n} \overline{f(y - x)} g(y) dy = \int_{\mathbb{R}^n} \overline{f(y)} g(y + x) dy = (f \star g)(x).$$

Por tanto, $(f \star g)$ está definida c.p.d en $\mathbb{R}^n$ y cumple la igualdad indicada. $\square$

En la segunda parte de la memoria, proporcionaremos una expresión equivalente para el marco discreto. En este, se dice que la convolución entre un núcleo (filtro) k y una imagen I es realizar la correlación con el núcleo invertido $\widetilde{k} * I$ (ya que la conjugación no es necesaria, al trabajar con números reales). Es interesante conocer la naturaleza de estas dos operaciones ya que, aunque una se pueda calcular a partir de la otra, son en esencia distintas, y tienen aplicaciones diferentes. De hecho, algunas librerías implementan la correlación y la denominan convolución y es el usuario el que debe de consultar la documentación y conocer esta distinción. En cierta medida, la correlación es un indicador de la similitud entre dos señales, comúnmente empleado para descubrir características relevantes en una señal desconocida al compararla con otra señal conocida. Es usado por tanto en reconocimiento de patrones. Sin embargo, la correlación no es en general una operación conmutativa ni asociativa y además no nos proporciona una interpretación tan clara en términos de frecuencia como la convolución. Por lo tanto, si queremos realizar una operación y “pensarla” en el dominio de la frecuencia usaremos la convolución.

5.4. Teorema de Convolución de Young

En esta sección, analizaremos el Teorema de Convolución de Young, que nos brinda un marco general en el que está definida la convolución. Antes de adentrarnos en el teorema en sí, examinaremos algunos casos particulares que son de especial interés y que serán fundamentales en múltiples ocasiones.

Teorema 5.4.1. Sean $1 \leq p, q \leq \infty$ tales que $\frac{1}{p} + \frac{1}{q} = 1$, y sean también $f \in \mathcal{L}^p(\mathbb{R}^n)$, $g \in \mathcal{L}^q(\mathbb{R}^n)$. Entonces se tiene que

1. $f * g$ está definida en todo $\mathbb{R}^n$.
2. $f * g$ es uniformemente continua en $\mathbb{R}^n$ y está acotada en $\mathbb{R}^n$.
3. Se cumple la desigualdad

$$\|f * g\|_\infty \leq \|f\|_p \|g\|_q.$$

Demostración. Como el lector puede imaginar, la primera parte de la prueba se sigue de una aplicación directa de la desigualdad de Hölder, la cual nos asegura que, dado $x \in \mathbb{R}^n$,

$$|(f * g)(x)| \leq \int_{\mathbb{R}^n} |f(x - y)g(y)| dy \leq \|f\|_p \|g\|_q.$$

Lo que prueba que $f * g$ está definida en todo $\mathbb{R}^n$, acotada en $\mathbb{R}^n$, y se cumple la desigualdad $\|f * g\|_\infty \leq \|f\|_p \|g\|_q$.

Faltaría probar que $f * g$ es uniformemente continua en $\mathbb{R}^n$. Suponemos para ello $p \neq \infty$. Sean $x, y \in \mathbb{R}^n$. Usando la desigualdad de Hölder de nuevo, se tiene que

$$\begin{aligned} |(f * g)(x) - (f * g)(t)| &\leq \int_{\mathbb{R}^n} |f(x - y) - f(t - y)| |g(y)| dy \\ &\leq \left(\int_{\mathbb{R}^n} |f(x - y) - f(t - y)|^p dy \right)^{1/p} \left(\int_{\mathbb{R}^n} |g(y)|^q dy \right)^{1/q} \\ &= \left(\int_{\mathbb{R}^n} |f(s) - f(s + t - x)|^p ds \right)^{1/p} \left(\int_{\mathbb{R}^n} |g(y)|^q dy \right)^{1/q} \\ &= w_p f(t - x) \|g\|_q. \end{aligned}$$

Finalmente, como

$$\lim_{t \rightarrow 0} w_p f(t) = 0,$$

se obtiene la continuidad uniforme de $f * g$ en $\mathbb{R}^n$. El caso de $p = \infty$, se sigue de aplicar el procedimiento anterior a $g * f$, ya que en ese caso $q = 1$. Y por el primer apartado de la proposición 5.3.2, $f * g = g * f$. $\square$

Observación 5.4.2. Este resultado es especialmente interesante, ya que nos da una definición de la convolución, para todo punto en $\mathbb{R}^n$, que además es uniformemente continua. Nótese que una de las aplicaciones más directas pudiera ser si una de las funciones está en $\mathcal{L}^\infty(\mathbb{R}^n)$, en ese caso si la otra fuera integrable en $\mathbb{R}^n$ automáticamente podríamos aplicar el resultado.

Teorema 5.4.3. Sean $1 \leq p \leq \infty$ y $f \in \mathcal{L}^1(\mathbb{R}^n)$, $g \in \mathcal{L}^p(\mathbb{R}^n)$. Entonces se tiene que

CAPÍTULO 5. CONVOLUCIÓN

1. $f * g$ está definida por doquier en $\mathbb{R}^n$.
2. $f * g \in \mathcal{L}^p(\mathbb{R}^n)$ y además se cumple la desigualdad

$$\|f * g\|_p \leq \|f\|_1 \|g\|_p.$$

Demostración. Para el caso en el que $p = \infty$, el Teorema 5.4.1 proporciona el resultado. Estamos buscando poder aplicar la desigualdad de Hölder. Para ello, consideramos p^* conjugado de p ($\frac{1}{p} + \frac{1}{p^*} = 1$). Dado $x \in \mathbb{R}^n$,

$$\begin{aligned} \int_{\mathbb{R}^n} |f(x-y)g(y)| dy &= \int_{\mathbb{R}^n} |f(x-y)|^{\frac{1}{p}} |f(x-y)|^{\frac{1}{p^*}} |g(y)| dy \\ &\leq \left(\int_{\mathbb{R}^n} |f(x-y)| dy \right)^{\frac{1}{p^*}} \left(\int_{\mathbb{R}^n} |f(x-y)||g(y)|^p dy \right)^{\frac{1}{p}}, \end{aligned}$$

de manera que se tendrá que

$$\begin{aligned} \int_{\mathbb{R}^n} \left(\int_{\mathbb{R}^n} |f(x-y)g(y)| dy \right)^p dx &\leq \int_{\mathbb{R}^n} \left(\int_{\mathbb{R}^n} |f(x-y)| dy \right)^{\frac{p}{p^*}} \left(\int_{\mathbb{R}^n} |f(x-y)||g(y)|^p dy \right) dx \\ &= \|f\|_1^{\frac{p}{p^*}} \int_{\mathbb{R}^n} \int_{\mathbb{R}^n} |f(x-y)||g(y)|^p dy dx \\ &= \|f\|_1^{\frac{p}{p^*}} \|f\|_1 \|g\|_p^p \\ &= \|f\|_1^p \|g\|_p^p < \infty, \end{aligned}$$

donde en la última igualdad se ha usado que

$$\frac{1}{p} + \frac{1}{p^*} = 1 \implies \frac{p+p^*}{pp^*} = 1 \implies \frac{p}{p^*} + 1 = p$$

Hemos obtenido por tanto que

$$\int_{\mathbb{R}^n} \left(\int_{\mathbb{R}^n} |f(x-y)g(y)| dy \right)^p dx < \infty.$$

Luego como consecuencia, $f * g$ está definida por doquier en $\mathbb{R}^n$ y $f * g \in \mathcal{L}^p(\mathbb{R}^n)$.

Para la última aseveración, basta con usar lo que acabamos de probar:

$$\|f * g\|_p^p = \int_{\mathbb{R}^n} \left| \int_{\mathbb{R}^n} |f(x-y)g(y)| dy \right|^p \leq \int_{\mathbb{R}^n} \left(\int_{\mathbb{R}^n} |f(x-y)g(y)| dy \right)^p \leq \|f\|_1^p \|g\|_p^p.$$

De aquí se obtiene que

$$\|f * g\|_p \leq \|f\|_1 \|g\|_p. \quad \square$$

Probamos finalmente el Teorema de Convolución de Young.

Teorema 5.4.4 (de convolución de Young). Sean $1 \leq p, q \leq \infty$ tales que $\frac{1}{p} + \frac{1}{q} = 1 + \frac{1}{r}$ y sean $f \in \mathcal{L}^p(\mathbb{R}^n), g \in \mathcal{L}^q(\mathbb{R}^n)$. Entonces se tiene que

5.4. TEOREMA DE CONVOLUCIÓN DE YOUNG

1. $f * g$ está definida c.p.d. en $\mathbb{R}^n$.
2. $f * g \in \mathcal{L}^r(\mathbb{R}^n)$ y además se cumple la desigualdad

$$\|f * g\|_r \leq \|f\|_p \|g\|_q.$$

Demostración. Comenzamos analizando los siguientes casos

- Si $p = \infty$, entonces se tiene que $\frac{1}{q} = 1 + \frac{1}{r} \leq 1$, lo cual implica que $r = \infty$ y $q = 1$. Por tanto, el Teorema 5.1.4 establece el resultado. De manera análoga ocurre si $q = \infty$.
- Si $p = 1$, entonces se tiene que $1 + \frac{1}{q} = 1 + \frac{1}{r} \leq 1$, lo cual implica que $r = q$. Por tanto, el Teorema 5.4.3 establece el resultado. De manera análoga ocurre si $q = 1$.

Tras estas consideraciones nos reduciremos al caso $1 < p, q < \infty$ y como consecuencia se tendrá que $1 < r < \infty$.

Para empezar, observamos que

$$\frac{1}{r} + \frac{r-p}{rp} + \frac{r-q}{rq} = \frac{1}{r} + \frac{1}{p} - \frac{1}{r} + \frac{1}{q} - \frac{1}{r} = \frac{1}{p} + \frac{1}{q} - \frac{1}{r} = 1.$$

Además, sabemos que:

- La función $y \mapsto (|f(x-y)|^p |g(y)|^q)^{\frac{1}{r}}$ es integrable en $\mathcal{L}^r(\mathbb{R}^n)$.
En efecto, como $f \in \mathcal{L}^p(\mathbb{R}^n)$ y $g \in \mathcal{L}^q(\mathbb{R}^n)$, tenemos que $|f|^p \in \mathcal{L}^1(\mathbb{R}^n)$ y $|g|^q \in \mathcal{L}^1(\mathbb{R}^n)$. Luego el Teorema 5.1.4 afirma que la convolución $|f|^p * |g|^q$ está definida c.p.d. en $\mathbb{R}^n$ y se cumple la desigualdad

$$\| |f|^p * |g|^q \|_1 \leq \|f\|_p^p \|g\|_q^q.$$

Y en consecuencia, la función $y \mapsto (|f(x-y)|^p |g(y)|^q)$ es integrable $\in \mathcal{L}^1(\mathbb{R}^n)$. De aquí se sigue que la función $y \mapsto (|f(x-y)|^p |g(y)|^q)^{\frac{1}{r}}$ es integrable $\in \mathcal{L}^r(\mathbb{R}^n)$.

- La función $y \mapsto |f(x-y)|^{\frac{r-p}{r}}$ es integrable en $\mathcal{L}^{\frac{pr}{r-p}}(\mathbb{R}^n)$. Esto se sigue de que la función $y \mapsto f(x-y)$ es integrable en $\mathcal{L}^p(\mathbb{R}^n)$.
- La función $y \mapsto |g(y)|^{\frac{r-q}{r}}$ es integrable en $\mathcal{L}^{\frac{qr}{r-q}}(\mathbb{R}^n)$. Esto se sigue de que la función $y \mapsto g(y)$ es integrable en $\mathcal{L}^q(\mathbb{R}^n)$.

Tomamos $x \in \mathbb{R}^n$, donde la convolución $|f| * |g|$ esté definida. Tenemos que

$$\begin{aligned} \int_{\mathbb{R}^n} |f(x-y)g(y)| dy &= \int_{\mathbb{R}^n} |f(x-y)|^{1+\frac{p}{r}-\frac{p}{r}} |g(y)|^{1+\frac{q}{r}-\frac{q}{r}} dy \\ &= \int_{\mathbb{R}^n} |f(x-y)|^{\frac{p}{r}} |g(y)|^{\frac{q}{r}} |f(x-y)|^{1-\frac{p}{r}} |g(y)|^{1-\frac{q}{r}} dy \\ &= \int_{\mathbb{R}^n} (|f(x-y)|^p |g(y)|^q)^{\frac{1}{r}} |f(x-y)|^{\frac{r-p}{r}} |g(y)|^{\frac{r-q}{r}} dy. \end{aligned} \quad (5.7)$$

Aplicando ahora la desigualdad de Hölder a (5.4),

$$\begin{aligned} &\int_{\mathbb{R}^n} (|f(x-y)|^p |g(y)|^q)^{\frac{1}{r}} |f(x-y)|^{\frac{r-p}{r}} |g(y)|^{\frac{r-q}{r}} dy \\ &\leq \left(\int_{\mathbb{R}^n} \left[(|f(x-y)|^p |g(y)|^q)^{\frac{1}{r}} \right]^r dy \right)^{\frac{1}{r}} \left(\int_{\mathbb{R}^n} |f(x-y)|^{\frac{r-p}{r} \frac{pr}{r-p}} dy \right)^{\frac{r-p}{pr}} \left(\int_{\mathbb{R}^n} |g(y)|^{\frac{r-q}{r} \frac{qr}{r-q}} dy \right)^{\frac{r-q}{qr}}. \end{aligned}$$

De aquí se sigue que

$$\int_{\mathbb{R}^n} (|f(x-y)|^p |g(y)|^q)^{\frac{1}{r}} |f(x-y)|^{\frac{r-p}{r}} |g(y)|^{\frac{r-q}{r}} dy \leq ((|f|^p * |g|^q)(x))^{\frac{1}{r}} \|f\|_p^{\frac{r-p}{r}} \|g\|_q^{\frac{r-q}{r}} < \infty.$$

Luego deducimos que $f * g$ está definida c.p.d. en $\mathbb{R}^n$, y que:

$$\left(\int_{\mathbb{R}^n} |f(x-y)g(y)| dy \right)^r \leq (|f|^p * |g|^q)(x) \|f\|_p^{r-p} \|g\|_q^{r-q}.$$

Finalmente, se tiene que

$$\begin{aligned} \int_{\mathbb{R}^n} |(f * g)(x)|^r dx &= \int_{\mathbb{R}^n} \left| \int_{\mathbb{R}^n} f(x-y)g(y) dy \right|^r dx \\ &\leq \int_{\mathbb{R}^n} \left(\int_{\mathbb{R}^n} |f(x-y)g(y)| dy \right)^r dx \leq \int_{\mathbb{R}^n} (|f|^p * |g|^q)(x) \|f\|_p^{r-p} \|g\|_q^{r-q} dx \\ &= \|f\|_p^{r-p} \|g\|_q^{r-q} \int_{\mathbb{R}^n} (|f|^p * |g|^q)(x) dx \leq \|f\|_p^{r-p} \|g\|_q^{r-q} \|f\|^p * |g|^q\|_1 \\ &\leq \|f\|_p^{r-p} \|g\|_q^{r-q} \|f\|_p^p \|g\|_q^q = \|f\|_p^r \|g\|_q^r. \end{aligned}$$

Esto demuestra que $f * g \in \mathcal{L}^r(\mathbb{R}^n)$, y además se tiene que $\|f * g\|_r \leq \|f\|_p \|g\|_q$. $\square$

5.5. Derivabilidad de la Convolución

Nace de manera natural preguntarse acerca de la derivabilidad de la convolución. En este sentido, presentaremos un teorema que establece condiciones bajo las cuales podemos derivar la convolución. Además, proporcionaremos una intuición sobre las potenciales aplicaciones en el ámbito de la computación de este resultado.

Teorema 5.5.1. *Sea $f \in \mathcal{L}^p(\mathbb{R}^n)$, $g \in \mathcal{L}^q(\mathbb{R}^n)$ con $1 \leq p, q \leq \infty$ tales que $\frac{1}{p} + \frac{1}{q} = 1$ y $k \in \{1, \dots, n\}$ de modo que $\frac{\partial f}{\partial x_k}$ está acotada. Entonces se verifica que*

$$\frac{\partial(f * g)}{\partial x_k}(x) = \left(\frac{\partial f}{\partial x_k} * g \right)(x) \quad \forall x \in \mathbb{R}^n. \quad (5.8)$$

Demostración. Como $f \in \mathcal{L}^p(\mathbb{R}^n)$, $g \in \mathcal{L}^q(\mathbb{R}^n)$ con $1 \leq p, q \leq \infty$ tales que $\frac{1}{p} + \frac{1}{q} = 1$, sabemos que $f * g$ está definida en todo $\mathbb{R}^n$, y es uniformemente continua en $\mathbb{R}^n$. Por otro lado, como $\frac{\partial f}{\partial x_k}$ está acotada, $\exists M \in \mathbb{R}^+$ tal que

$$\left| \frac{\partial f}{\partial x_k}(x) \right| \leq M \quad \forall x \in \mathbb{R}^n.$$

Sea la función $\gamma : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{C}$ definida por

$$\gamma(x, y) = f(x-y)g(y).$$

Se tiene que γ verifica los siguientes puntos:

- es integrable con respecto la variable y .
- es derivable con respecto a la variable x_k y se tiene que:

$$\frac{\partial \gamma}{\partial x_k}(x, y) = \frac{\partial (f(x-y)g(y))}{\partial x_k} = \frac{\partial f(x-y)}{\partial x_k} g(y),$$

y además

$$\left| \frac{\partial f}{\partial x_k}(x-y)g(y) \right| \leq M|g(y)| \quad \forall (x, y) \in \mathbb{R}^n \times \mathbb{R}^n.$$

Y, como, $g \in \mathcal{L}^1(\mathbb{R}^n)$, $\frac{\partial f}{\partial x_k}(x-y)g(y)$ es integrable $\forall x \in \mathbb{R}^n$.

Tomamos $x = (x_1, \dots, x_n) \in \mathbb{R}^n$. El Teorema de derivación de integrales dependientes de un parámetro asegura que $f * g$ es derivable con respecto a x_k , y también:

$$\begin{aligned} \frac{\partial (f * g)}{\partial x_k}(x) &= \int_{\mathbb{R}^n} \frac{\partial}{\partial x_k} (f(x-y)g(y)) dy \\ &= \int_{\mathbb{R}^n} \frac{\partial f(x-y)}{\partial x_k} g(y) dy = \left(\frac{\partial f}{\partial x_k} * g \right)(x). \end{aligned}$$

De donde concluimos la fórmula (5.5.1). □

Teorema 5.5.2. Sean $1 \leq p, q \leq \infty$ tales que $\frac{1}{p} + \frac{1}{q} = 1$ y sean $f \in \mathcal{L}^p(\mathbb{R}^n)$, $g \in \mathcal{L}^q(\mathbb{R}^n)$. Suponemos que $\alpha \in (\mathbb{N} \cup \{0\})^n$ cumple que para cada $\beta \in A_\alpha$ la función $D_\beta f \in \mathcal{L}^p(\mathbb{R}^n)$ y está acotada. Entonces

$$D_\alpha(f * g)(x) = (D_\alpha(f) * g)(x) \quad x \in \mathbb{R}^n.$$

Demostración. Para un $\alpha \in (\mathbb{N} \cup \{0\})^n$ cualquiera, basta con usar el Teorema 5.5.1 y realizar inducción sobre $|\alpha|$. □

Este resultado muestra la propiedad regularizadora que tiene la convolución. En efecto, nos permite derivar el término regular. En el ámbito computacional, esta propiedad se utiliza con cierta frecuencia.

Ilustramos un ejemplo completo para mostrar al lector una de las diversas aplicaciones en VC, que reúne todos los conceptos que hemos ido tratando a lo largo de este capítulo.

Ejemplo 5.5.3. En ocasiones, surge la necesidad de identificar características específicas en las imágenes, como los bordes. Por ejemplo, si queremos encontrar un determinado objeto en una imagen, pudiera ser interesante limitar las distintas regiones.

En términos abstractos, los bordes representan transiciones abruptas o cambios rápidos en la intensidad de una imagen. Podemos definir matemáticamente una imagen como una función I , tal que $I(x, y)$ representa la intensidad en una posición específica (x, y) de la imagen. La derivada, que mide la tasa de cambio o variación en el valor de esta función, nos permite identificar áreas en la imagen donde la intensidad cambia rápidamente. Para evitar que se produzca ruido en la imagen, primero trataremos de suavizar la imagen usando algún filtro gaussiano G_σ . Por tanto, querríamos saber calcular $\frac{\partial}{\partial x}(I * G_\sigma)$, $\frac{\partial}{\partial y}(I * G_\sigma)$.

Aplicando el Teorema de derivación de la convolución para computar estas derivadas, necesitamos únicamente computar la derivada de la función gaussiana (que es regular). Por lo que

CAPÍTULO 5. CONVOLUCIÓN

para derivar cualquier imagen solo tendremos que convolucionarla con la derivada del núcleo, teniendo así una manera genérica y eficiente de realizar operaciones de derivación de imágenes.

Sabiendo calcular estas derivadas podemos estar interesados en conocer también la magnitud del gradiente $\sqrt{\left(\frac{\partial}{\partial x}(I * G_\sigma)\right)^2 + \left(\frac{\partial}{\partial y}(I * G_\sigma)\right)^2}$, la orientación del gradiente $\arctan\left(\frac{\frac{\partial}{\partial y}(I * G_\sigma)}{\frac{\partial}{\partial x}(I * G_\sigma)}\right)$, o el Laplaciano $\left(\frac{\partial^2}{\partial x^2}(I * G_\sigma) + \frac{\partial^2}{\partial y^2}(I * G_\sigma)\right)$ de la imagen suavizada. Este último podemos verlo como un enfoque global de las contribuciones de las segundas derivadas. Evidentemente, en todos estos cálculos seguimos convolucionando la imagen con las derivadas del núcleo Gaussiano.

Observación 5.5.4. En la figura 5.1 se muestra el resultado de estas operaciones sobre una determinada imagen. La implementación de todas las convoluciones mencionadas se lleva a cabo con una función personalizada cuya implementación está basada en el Teorema de Convolución. Esto implica calcular la Transformada de Fourier de los elementos involucrados, realizar el producto punto a punto entre ellos, y finalmente aplicar la Transformada de Fourier inversa al resultado.

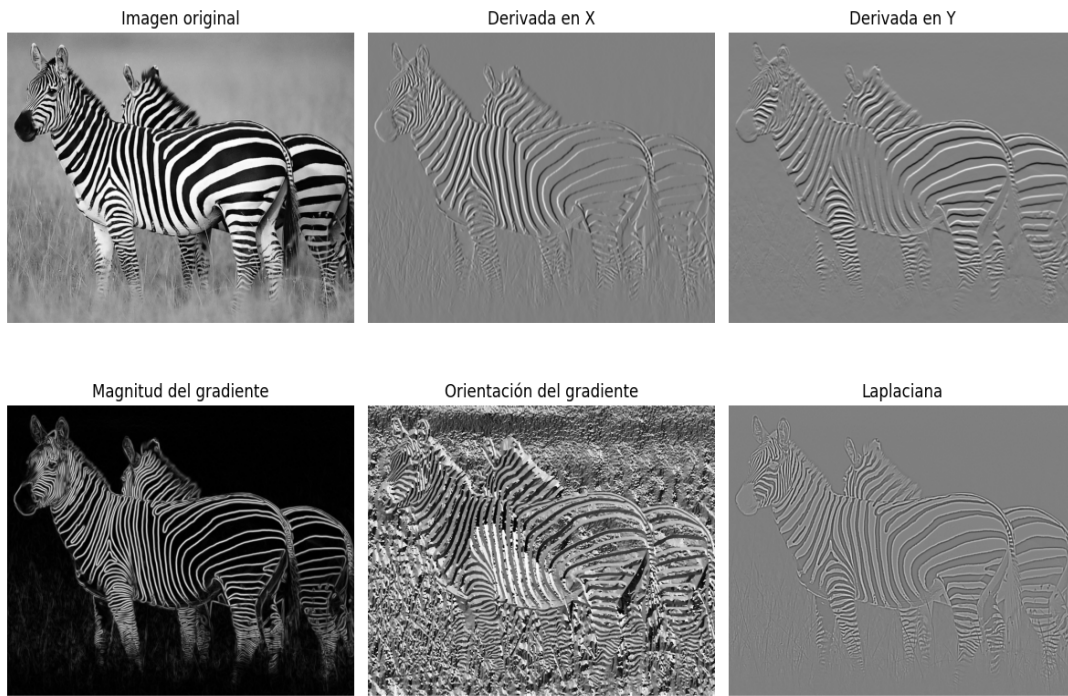


Figura 5.1.: Resultado de la ejecución de la derivada en X, derivada en Y, magnitud del gradiente, orientación del gradiente y el Laplaciano de la imagen original

En efecto, atendiendo a dicha figura, comprobamos que:

- Cuando derivamos con respecto a $\mathbf{x}$, se realzan los bordes verticales ya que estamos buscando cambios abruptos de intensidad en el eje $\mathbf{x}$. De manera análoga ocurre cuando derivamos con respecto a $\mathbf{y}$, ya que lo que estamos haciendo es medir la tasa de variación

de la intensidad en este eje detectando por tanto los bordes horizontales. Los bordes pueden ser blancos o negros, dependiendo de si ha habido un aumento o una disminución de intensidad.

- La magnitud del gradiente engloba ambas derivadas, y por eso podemos detectar tanto los bordes verticales como horizontales.
- La orientación del gradiente nos muestra información sobre la dirección en la que cambia la intensidad de la imagen.
- El Laplaciano combina contribuciones de ambas derivadas de segundo orden, lo que facilita la detección de bordes tanto verticales como horizontales. Esto lo convierte en una excelente opción para la detección de características.

5.6. Núcleos de Sumabilidad

La proposición 5.3.2 afirma que $\mathcal{L}^1(\mathbb{R}^n)$ tiene estructura de anillo con respecto a las operaciones $+$ y $*$. Sin embargo, debemos notar que $\mathcal{L}^1(\mathbb{R}^n)$ carece de unidad.

En efecto, supongamos que exista $K \in \mathcal{L}^1(\mathbb{R}^n)$ tal que se verificara

$$K * f = f * K = f \quad \forall f \in \mathcal{L}^1(\mathbb{R}^n).$$

Entonces, tomando $f = G \in \mathcal{L}^1(\mathbb{R}^n)$, tenemos

$$\widehat{K}(y)G(y) = \widehat{K}(y)\widehat{G}(y) = G(y) \quad \forall y \in \mathbb{R}^n,$$

de donde deducimos que necesariamente $\widehat{K}(y) = 1 \quad \forall y \in \mathbb{R}^n$. Sin embargo, esto contradice el Lema de Riemann-Lebesgue.

Esta irregularidad no se presenta en el contexto discreto. Estudiaremos este hecho con mayor detenimiento en la segunda parte de la memoria. En ella, analizaremos el filtro impulso, que será aquel que, convolucionándolo con cualquier señal, resulte en ella misma. Vale la pena mencionar, en relación al análisis realizado en el capítulo anterior acerca de las operaciones de correlación y convolución, que este fenómeno no se manifiesta con la operación de correlación, marcando así una clara distinción entre ambas técnicas, y subrayando la convolución como una operación más intuitiva en términos de frecuencia, ya que proporciona una salida más natural.

No obstante, $\mathcal{L}^1(\mathbb{R}^n)$ cuenta con unidades aproximadas, denominadas núcleos de sumabilidad, que tratan de paliar los indeseados efectos provocados por la ausencia de unidad.

Previo a establecer de forma adecuadamente amplia el concepto de núcleo de sumabilidad, introducimos algunas definiciones.

Definición 5.6.1. Un *conjunto dirigido* es un conjunto Λ provisto de un orden $\leq$, tal que para cualesquiera $\lambda_1, \lambda_2 \in \Lambda$ existe $\mu \in \Lambda$ con $\lambda_1 \leq \mu$ y $\lambda_2 \leq \mu$.

Definición 5.6.2. Una *red* en un espacio topológico X es una familia $\{x_\lambda\}_{\lambda \in \Lambda}$, donde Λ es un conjunto dirigido.

Se dice que la red es convergente si existe $x \in X$ tal que para cada entorno U de x existe $\mu \in \Lambda$ tal que

$$\mu \in \Lambda, \mu \leq \lambda \implies x_\lambda \in U.$$

CAPÍTULO 5. CONVOLUCIÓN

Si X es un espacio Hausdorff, entonces el punto x es necesariamente único y se denota por $\lim_{\lambda \in \Lambda}$.

Ejemplo 5.6.3. Veamos algunos ejemplos para familiarizarnos con estas definiciones.

- Tomamos como conjunto dirigido $\mathbb{N}_0$ con orden creciente, en cuyo caso

$$\lim_{\lambda \in \Lambda} = \lim_{n \rightarrow \infty}.$$

- Tomamos como conjunto dirigido $\mathbb{R}^+$ con orden decreciente, en cuyo caso

$$\lim_{\lambda \in \Lambda} = \lim_{\substack{t \rightarrow 0 \\ t > 0}}.$$

Definimos finalmente el concepto de núcleo de sumabilidad $\{K_\lambda\}_{\lambda \in \Lambda}$, que como puede intuir el lector poseerá la cualidad

$$\lim_{\lambda \in \Lambda} \|f * K - f\|_1 = 0 \quad \forall f \in \mathcal{L}^1(\mathbb{R}^n).$$

Definición 5.6.4. Un *núcleo de sumabilidad* es una red $\{K_\lambda\}_{\lambda \in \Lambda}$ de funciones de $\mathcal{L}^1(\mathbb{R}^n)$ que verifica lo siguiente:

- para cada $\lambda \in \Lambda$,

$$\int_{\mathbb{R}^n} K_\lambda(x) dx = 1;$$

- existe una constante $M \in \mathbb{R}^+$ tal que, para cada $\lambda \in \Lambda$,

$$\int_{\mathbb{R}^n} |K_\lambda(x)| dx \leq M;$$

- para cada $\delta \in \mathbb{R}^+$, se tiene que

$$\lim_{\lambda \in \Lambda} \int_{||x|| > \delta} |K_\lambda(x)| dx = 0.$$

A continuación, presentamos un resultado que proporciona una amplia gama de núcleos de sumabilidad.

Proposición 5.6.5. Sea $K \in \mathcal{L}^1(\mathbb{R}^n)$ tal que

$$\int_{\mathbb{R}^n} K(x) dx = 1$$

y, para cada $\epsilon \in \mathbb{R}^+$, se define $K_\epsilon : \mathbb{R} \rightarrow \mathbb{C}$ por

$$K_\epsilon(x) = \epsilon^{-n} K(\epsilon^{-1}x) \quad \forall x \in \mathbb{R}^n.$$

Entonces la familia $\{K_\epsilon\}_{\epsilon \in \mathbb{R}^+}$ es un núcleo de sumabilidad.

Demostración. Comprobaremos las condiciones que debe verificar un núcleo de sumabilidad.

- Para cada $\epsilon \in \mathbb{R}^+$, se tiene realizando un cambio de variable

$$\int_{\mathbb{R}^n} K_\epsilon(x) dx = \int_{\mathbb{R}^n} \epsilon^{-n} K(\epsilon^{-1}x) dx = \int_{\mathbb{R}^n} K(x) dx = 1.$$

- Para cada $\epsilon \in \mathbb{R}^+$, se tiene

$$\int_{\mathbb{R}^n} |K_\epsilon(x)| dx = \int_{\mathbb{R}^n} \epsilon^{-n} |K(\epsilon^{-1}x)| dx = \int_{\mathbb{R}^n} |K(x)| dx = \|K\|_1.$$

Por tanto, se verifica la segunda condición de la definición 5.6.4 con constante $M = \|K\|_1$.

- Fijamos un $\delta \in \mathbb{R}^+$, tenemos por tanto que

$$\int_{\|x\| > \delta} |K_\epsilon(x)| dx = \int_{\|x\| > \delta} \epsilon^{-n} |K(\epsilon^{-1}x)| dx = \int_{\|x\| > \delta/\epsilon} |K(x)| dx.$$

Debemos entonces probar que

$$\lim_{\substack{\epsilon \rightarrow 0 \\ \epsilon > 0}} \int_{\|x\| > \delta/\epsilon} |K(x)| dx = 0.$$

Para ello, vamos a usar el Teorema de la convergencia dominada. Sea $\{\epsilon_m\}$ una sucesión en $\mathbb{R}^+$ tal que $\{\epsilon_m\} \rightarrow 0$. Definimos una sucesión de funciones $\{\phi_m\} \in \mathcal{L}^1(\mathbb{R}^n)$ por

$$\phi_m(x) = \begin{cases} |K(x)| & \text{si } \|x\| > \delta/\epsilon_m, \\ 0 & \text{si } \|x\| < \delta/\epsilon_m. \end{cases}$$

Observamos que

$$\int_{\|x\| > \delta/\epsilon_m} |K(x)| dx = \int_{\mathbb{R}^n} \phi_m(x) dx \quad \forall m \in \mathbb{N}.$$

Aplicamos entonces el Teorema de la convergencia dominada a ϕ_m .

- Como $\{\delta/\epsilon_m\} \rightarrow 0$,

$$\lim_{m \rightarrow \infty} \phi_m(x) = 0 \quad \forall x \in \mathbb{R}^n.$$

- Se tiene que

$$|\phi_m(x)| \leq |K(x)| \quad \forall x \in \mathbb{R}^n, \forall m \in \mathbb{N},$$

con $|K|$ integrable en $\mathbb{R}^n$.

Luego el Teorema de la convergencia dominada garantiza que

$$\lim_{m \rightarrow \infty} \int_{\mathbb{R}^n} \phi_m(x) dx = 0.$$

De aquí deducimos que

$$\lim_{m \rightarrow \infty} \int_{\|x\| > \delta/\epsilon_m} |K(x)| dx = 0,$$

cumpliéndose así la última condición.

Concluimos que para cada $\epsilon \in \mathbb{R}^+$ la familia $\{K_\epsilon\}_{\epsilon \in \mathbb{R}^+}$ es un núcleo de sumabilidad. $\square$

Ejemplo 5.6.6 (Núcleo de Gauss-Weierstrass). La familia de funciones $\{G_t\}_{t \downarrow 0}$ definida por

$$G_t : \mathbb{R}^n \rightarrow \mathbb{R}, G_t(x) = \frac{1}{(4\pi t)^{-n/2}} e^{-\pi \|x\|^2} \quad \forall x \in \mathbb{R}^n, \forall t \in \mathbb{R}^+,$$

se denomina *núcleo de Gauss-Weierstrass*.

Observamos que este núcleo es la familia de funciones $\{K_{\sqrt{4\pi t}}\}_{t \in \mathbb{R}^+}$ asociada a la proposición 5.6.5, con $K = G \in \mathcal{L}^1(\mathbb{R}^n)$.

Definición 5.6.7. Definimos $P : \mathbb{R}^n \rightarrow \mathbb{R}$ función de $\mathcal{L}^1(\mathbb{R}^n)$ dada por

$$P(x) = C_n \frac{1}{(1 + \|x\|^2)^{\frac{n+1}{2}}}, \quad \text{con } C_n = \frac{\Gamma(\frac{n+1}{2})}{\pi^{\frac{n+1}{2}}}$$

y donde Γ es la función gamma.

Lema 5.6.8. P cumple

$$\int_{\mathbb{R}^n} P(x) dx = 1. \quad (5.9)$$

Demostración. Comprobaremos que se verifica

$$\frac{1}{C_n} = \int_{\mathbb{R}^n} \frac{1}{(1 + \|x\|^2)^{\frac{n+1}{2}}} dx,$$

que es equivalente a (5.6.8).

Usamos coordenadas polares:

$$\int_{\mathbb{R}^n} \frac{1}{(1 + \|x\|^2)^{\frac{n+1}{2}}} dx = w_{n-1} \int_0^\infty \frac{r^{n-1}}{(1 + r^2)^{\frac{n+1}{2}}} dr,$$

donde w_{n-1} denota el área de $\mathbb{S}^{n-1}$.

Aplicamos el cambio de variable, $r = \tan(\theta)$, y obtenemos

$$\begin{aligned} w_{n-1} \int_0^\infty \frac{r^{n-1}}{(1 + r^2)^{\frac{n+1}{2}}} dr &= w_{n-1} \int_0^{\pi/2} \frac{(\tan(\theta))^{n-1}}{(1 + (\tan(\theta))^2)^{\frac{n+1}{2}}} (1 + \tan(\theta))^2 d\theta \\ &= w_{n-1} \int_0^{\pi/2} \frac{(\tan(\theta))^{n-1} (\cos(\theta))^{n+1}}{(\cos(\theta))^2} d\theta \\ &= w_{n-1} \int_0^{\pi/2} (\sin(\theta))^{n-1} d\theta. \end{aligned}$$

Realizando el cambio de variable $u = (\sin(\theta))^2$, resulta

$$w_{n-1} \int_0^{\pi/2} (\sin(\theta))^{n-1} d\theta = \frac{1}{2} w_{n-1} \int_0^1 (1-s)^{\frac{n}{2}-1} s^{\frac{1}{2}-1} ds = \frac{1}{2} w_{n-1} \beta\left(\frac{n}{2}, \frac{1}{2}\right).$$

Sustituyendo ahora el valor de w_{n-1} y utilizando la relación de la función beta β con la función gamma Γ , se tiene:

$$w_{n-1}\beta\left(\frac{n}{2}, \frac{1}{2}\right) = \frac{2\Gamma\left(\frac{n}{2}\right)\Gamma\left(\frac{1}{2}\right)}{\Gamma\left(\frac{n+1}{2}\right)} = \frac{\pi^{\frac{n+1}{2}}}{\Gamma\left(\frac{n+1}{2}\right)}.$$

Concluimos que

$$\int_{\mathbb{R}^n} \frac{1}{(1 + \|x\|^2)^{\frac{n+1}{2}}} dx = \frac{\pi^{\frac{n+1}{2}}}{\Gamma\left(\frac{n+1}{2}\right)} = \frac{1}{C_n}. \quad \square$$

Ejemplo 5.6.9 (Núcleo de Poisson). La familia de funciones $\{P_y\}_{y>0}$ definida por

$$P_y : \mathbb{R}^n \rightarrow \mathbb{R}, \quad P_y(x) = C_n \frac{y}{(y^2 + \|x\|^2)^{\frac{n+1}{2}}}, \quad \text{con } C_n = \frac{\Gamma\left(\frac{n+1}{2}\right)}{\pi^{\frac{n+1}{2}}},$$

se denomina *núcleo de Poisson*.

Observamos que este núcleo es la familia de funciones $\{K_y\}_{y \in \mathbb{R}^+}$ asociada a la proposición 5.6.5 con $K = P \in \mathcal{L}^1(\mathbb{R}^n)$.

Lema 5.6.10. Sea $K \in \mathcal{L}^1(\mathbb{R}^n)$ tal que

$$\int_{\mathbb{R}^n} K(x) dx = 1.$$

Supongamos $1 \leq p < \infty$, y $f \in \mathcal{L}^p(\mathbb{R}^n)$. Entonces

$$\|f * K_\lambda - f\|_p \leq \int_{\mathbb{R}^n} w_p f(x) |K_\lambda(x)| dx.$$

Demostración. Para cada $x \in \mathbb{R}^n$ tal que $(f * K)(x)$ está definida, tenemos

$$\begin{aligned} |(f * K)(x) - f(x)| &= \left| \int_{\mathbb{R}^n} f(x-t) K(t) dt - f(x) \int_{\mathbb{R}^n} K(t) dt \right| \\ &\leq \int_{\mathbb{R}^n} |f(x-t) - f(x)| |K(t)| dt. \end{aligned}$$

Distinguiamos dos casos:

- Suponemos que $p = 1$. Usando el Teorema de Tonelli, tenemos que

$$\begin{aligned} \|f * K - f\|_1 &= \int_{\mathbb{R}^n} |(f * K_\lambda)(x) - f(x)| dx \\ &\leq \int_{\mathbb{R}^n} \int_{\mathbb{R}^n} |f(x-t) - f(x)| |K_\lambda(t)| dt dx \\ &= \int_{\mathbb{R}^n} \int_{\mathbb{R}^n} |f(x-t) - f(x)| |K(t)| dx dt = \int_{\mathbb{R}^n} w_1 f(t) |K(t)| dt. \end{aligned}$$

- Suponemos que $1 < p < \infty$. Tomamos $q = \frac{p}{p-1}$, de modo que $\frac{1}{p} + \frac{1}{q} = 1$. Aplicando el

Teorema de Tonelli de nuevo, obtenemos

$$\begin{aligned}
 \|f * K - f\|_p^p &= \int_{\mathbb{R}^n} |(f * K_\lambda)(x) - f(x)|^p dx = \\
 &\leq \int_{\mathbb{R}^n} |(f * K_\lambda)(x) - f(x)| |(f * K_\lambda)(x) - f(x)|^{p-1} dx. \\
 &\leq \int_{\mathbb{R}^n} \left(\int_{\mathbb{R}^n} |f(x-t) - f(x)| |K(t)| dt \right) |(f * K_\lambda)(x) - f(x)|^{p-1} dx. \\
 &\leq \int_{\mathbb{R}^n} \left(\int_{\mathbb{R}^n} |f(x-t) - f(x)| |(f * K_\lambda)(x) - f(x)|^{p-1} dx \right) |K(t)| dt.
 \end{aligned}$$

Ahora aplicamos la desigualdad de Hölder

$$\begin{aligned}
 &\int_{\mathbb{R}^n} \left(\int_{\mathbb{R}^n} |f(x-t) - f(x)| |(f * K_\lambda)(x) - f(x)|^{p-1} dx \right) \\
 &\leq \left(\int_{\mathbb{R}^n} |f(x-t) - f(x)|^p dx \right)^{1/p} \left(\int_{\mathbb{R}^n} |(f * K_\lambda)(x) - f(x)|^{(p-1)q} dx \right)^{1/q} \\
 &\leq \left(\int_{\mathbb{R}^n} |f(x-t) - f(x)|^p dx \right)^{1/p} \left(\int_{\mathbb{R}^n} |(f * K_\lambda)(x) - f(x)|^p dx \right)^{1/q} \\
 &= w_p f(t) \|f * K - f\|_p^{p/q}.
 \end{aligned}$$

De donde deducimos que

$$\|f * K - f\|_p^p \leq \int_{\mathbb{R}^n} w_p f(t) \|f * K - f\|_p^{p/q} |K(t)| dt = \|f * K - f\|_p^{p/q} \int_{\mathbb{R}^n} w_p f(t) |K(t)| dt,$$

luego

$$\|f * K - f\|_p \leq \int_{\mathbb{R}^n} w_p f(t) |K(t)| dt.$$

En ambos casos se verifica la desigualdad. $\square$

Teorema 5.6.11. Sea $\{K_\lambda\}_{\lambda \in \Lambda}$ un núcleo de sumabilidad. Supongamos $1 \leq p < \infty$, y $f \in \mathcal{L}^p(\mathbb{R}^n)$. Entonces

$$\lim_{\lambda \in \Lambda} \|f * K_\lambda - f\|_p = 0.$$

Demostración. Fijamos $\delta \in \mathbb{R}^+$, para cada $\lambda \in \Lambda$, usando el Lema 5.6.10, se tiene que

$$\begin{aligned}
 \|f * K_\lambda - f\|_p &\leq \int_{\mathbb{R}^n} w_p f(x) |K_\lambda(x)| dx \\
 &= \int_{\|x\| < \delta} w_p f(x) |K_\lambda(x)| dx + \int_{\|x\| > \delta} w_p f(x) |K_\lambda(x)| dx.
 \end{aligned}$$

Analizamos cada uno de los sumandos por separado. Para el primero, usamos que K_λ es un

núcleo de sumabilidad:

$$\int_{||x|| < \delta} w_p f(x) |K_\lambda(x)| dx \leq \sup_{||x|| < \delta} w_p f(x) \int_{||x|| < \delta} |K_\lambda(x)| dx \leq M \sup_{||x|| < \delta} w_p f(x).$$

Y, por otro, usando la proposición 3.3.2, sabemos que

$$\int_{||x|| > \delta} w_p f(x) |K_\lambda(x)| dx \leq 2||f||_p \int_{||x|| > \delta} |K_\lambda(x)| dx.$$

Juntando ambas acotaciones,

$$||f * K_\lambda - f||_p \leq M \sup_{||x|| < \delta} w_p f(x) + 2||f||_p \int_{||x|| > \delta} |K_\lambda(x)| dx.$$

Luego

$$\limsup_{\lambda \in \Lambda} ||f * K_\lambda - f||_p \leq M \sup_{||x|| < \delta} w_p f(x).$$

Esto es válido para todo $\delta \in \mathbb{R}^+$, por lo que

$$\lim_{\delta \rightarrow 0} \sup_{||x|| < \delta} w_p f(x) = 0.$$

En consecuencia

$$\limsup_{\lambda \in \Lambda} ||f * K_\lambda - f||_p = 0,$$

concluyendo que

$$\lim_{\lambda \in \Lambda} ||f * K_\lambda - f||_p = 0. \quad \square$$

Corolario 5.6.12. Sea $K \in \mathcal{L}^1(\mathbb{R}^n)$ tal que

$$\int_{\mathbb{R}^n} K(x) dx = 1.$$

Para cada $\epsilon \in \mathbb{R}^+$, se define $K_\epsilon : \mathbb{R} \rightarrow \mathbb{C}$ por

$$K_\epsilon(x) = \epsilon^{-n} K(\epsilon^{-1}x) \quad \forall x \in \mathbb{R}^n.$$

Supongamos $1 \leq p < \infty$, y $f \in \mathcal{L}^p(\mathbb{R}^n)$. Entonces

$$\lim_{\substack{\epsilon \rightarrow 0 \\ \epsilon > 0}} ||f * K_\epsilon - f||_p = 0.$$

Demostración. Basta con usar el Teorema 5.6.11, y la Proposición 5.6.5. $\square$

Teorema 5.6.13. Sea $\{K_\lambda\}_{\lambda \in \Lambda}$ un núcleo de sumabilidad, y $f \in \mathcal{L}^\infty(\mathbb{R}^n)$.

- Supongamos que f es continua en cada punto de un subconjunto compacto $E \subset \mathbb{R}^n$.
Entonces

$$\lim_{\lambda \in \Lambda} \max_{x \in E} |(f * K_\lambda)(x) - f(x)| = 0.$$

CAPÍTULO 5. CONVOLUCIÓN

En particular, si f es continua en un punto $x \in \mathbb{R}^n$,

$$\lim_{\lambda \in \Lambda} (f * K_\lambda)(x) = f(x).$$

- Supongamos que f es uniformemente continua en $\mathbb{R}^n$. Entonces

$$\lim_{\lambda \in \Lambda} \|(f * K_\lambda)(x) - f(x)\|_\infty = 0.$$

Demostración. Como $f \in \mathcal{L}^\infty(\mathbb{R}^n)$, sabemos que la convolución $f * K_\lambda$ está definida en todo $\mathbb{R}^n$ y está acotada. Dado $\epsilon \in \mathbb{R}^+$, notemos los siguientes puntos:

- Por la continuidad de f en cada punto del conjunto compacto E , se tiene que $\exists \delta \in \mathbb{R}^+$ tal que

$$x \in E, \|x - t\| < \delta \implies |f(x - t) - f(x)| < \frac{\epsilon}{2M}.$$

- Al ser $\{K_\lambda\}_{\lambda \in \Lambda}$ un núcleo de sumabilidad, existe $M > 0$ tal que para todo $\lambda \in \Lambda$

$$\int_{\mathbb{R}^n} |K_\lambda(x)| dx \leq M.$$

Además, también existe $\mu \in \Lambda$ tal que

$$\lambda \in \Lambda, \mu \leq \lambda \implies 2\|f\|_\infty \int_{\|x\| > \delta} |K_\lambda(x)| dx < \frac{\epsilon}{2}.$$

Para $\lambda \in \Lambda$ con $\mu \leq \lambda$, y para cada $x \in E$, usando los puntos anteriores, se tiene que

$$\begin{aligned} |(f * K_\lambda)(x) - f(x)| &= \left| \int_{\mathbb{R}^n} f(x - t) K_\lambda(t) dt - f(x) \int_{\mathbb{R}^n} K_\lambda(t) dt \right| \\ &\leq \int_{\mathbb{R}^n} |f(x - t) - f(x)| |K_\lambda(t)| dt \\ &= \int_{\|t\| < \delta} |f(x - t) - f(x)| |K_\lambda(t)| dt + \int_{\|t\| > \delta} |f(x - t) - f(x)| |K_\lambda(t)| dt \\ &\leq \frac{\epsilon}{2M} \int_{\|t\| < \delta} |K_\lambda(t)| dt + 2\|f\|_\infty \int_{\|t\| > \delta} |K_\lambda(t)| dt \leq \frac{\epsilon}{2M} M + \frac{\epsilon}{2} = \epsilon. \end{aligned}$$

Por tanto,

$$\max_{x \in E} |(f * K_\lambda)(x) - f(x)| \leq \epsilon.$$

Finalmente, si f es continua en un punto $x \in \mathbb{R}^n$, tomando como conjunto compacto $E = \{x\}$, se sigue que

$$\lim_{\lambda \in \Lambda} (f * K_\lambda)(x) = f(x). \quad \square$$

Corolario 5.6.14. Sea $K \in \mathcal{L}^1(\mathbb{R}^n)$ tal que

$$\int_{\mathbb{R}^n} K(x) dx = 1.$$

Para cada $\epsilon \in \mathbb{R}^+$, se define $K_\epsilon : \mathbb{R} \rightarrow \mathbb{C}$ por

$$K_\epsilon(x) = \epsilon^{-n} K(\epsilon^{-1}x) \quad \forall x \in \mathbb{R}^n.$$

Sea también $f \in \mathcal{L}^\infty(\mathbb{R}^n)$.

- Supongamos que f es continua en cada punto de un subconjunto compacto $E \subset \mathbb{R}^n$. Entonces

$$\lim_{\substack{\epsilon \rightarrow 0 \\ \epsilon > 0}} \max_{x \in E} |(f * K_\epsilon)(x) - f(x)| = 0.$$

En particular, si f es continua en un punto $x \in \mathbb{R}^n$,

$$\lim_{\substack{\epsilon \rightarrow 0 \\ \epsilon > 0}} (f * K_\epsilon)(x) = f(x).$$

- Supongamos que f es uniformemente continua en $\mathbb{R}^n$. Entonces

$$\lim_{\substack{\epsilon \rightarrow 0 \\ \epsilon > 0}} \|(f * K_\epsilon)(x) - f(x)\|_\infty = 0.$$

Demostración. Basta con usar el Teorema 5.6.13, y la proposición 5.6.5. □

Observación 5.6.15. Cabe mencionar que en el segundo apartado del Teorema 5.6.13 y el Corolario 5.6.14 se exige que la función f sea uniformemente continua, y esta condición no es caprichosa. En efecto, como hemos anotado anteriormente, al estar $f \in \mathcal{L}^\infty(\mathbb{R}^n)$, se tiene que para todo $\lambda \in \Lambda$, $f * K_\lambda$ es uniformemente continua en $\mathbb{R}^n$, por lo que si se cumple que

$$\lim_{\lambda \in \Lambda} \|(f * K_\lambda)(x) - f(x)\|_\infty = 0,$$

se tiene que necesariamente f es uniformemente continua en $\mathbb{R}^n$.

5.7. Métodos de Sumación

Retomamos en esta sección el desafío de reconstruir una función $f \in \mathcal{L}^1(\mathbb{R}^n)$ a partir de su Transformada de Fourier $\hat{f}$. El lector puede pensar que este objetivo ya se alcanzó mediante el Teorema de Inversión. Sin embargo, para su aplicación, se obligaba a que necesariamente la función Transformada fuera integrable en $\mathbb{R}^n$ (de hecho, si no lo era, no tenía sentido la reconstrucción planteada en (4.8.1)). Sin embargo, como ya comentamos en el primer capítulo, esta es una hipótesis ciertamente restrictiva, y es por ello por lo que se plantean otras posibles interpretaciones para reconstruir la función.

Recordamos los métodos de sumabilidad de las series de Fourier, que consistían en la utilización

CAPÍTULO 5. CONVOLUCIÓN

de las sumas ponderadas

$$\sum_{k=-\infty}^{\infty} \widehat{f}(k) e^{ikx} \Phi(\epsilon k),$$

donde Φ era una función de peso adecuada y la ejecución del límite en $\epsilon = 0$ de éstas. Parece natural tratar de utilizar en nuestra situación integrales ponderadas

$$\int_{\mathbb{R}^n} \widehat{f}(y) e^{2\pi i \langle x, y \rangle} \Phi(\epsilon y) dy$$

y ejecutar el límite en $\epsilon = 0$.

Proposición 5.7.1. *Sea $\Phi \in \mathcal{L}^1(\mathbb{R}^n)$ tal que*

$$\Phi(-x) = \Phi(x) \quad \forall x \in \mathbb{R}^n,$$

y pongamos $K = \widehat{\Phi}$. Para cada $\epsilon \in \mathbb{R}^+$, se define la función $K_\epsilon : \mathbb{R} \rightarrow \mathbb{C}$ por

$$K_\epsilon(x) = \epsilon^{-n} K(\epsilon^{-1}x) \quad \forall x \in \mathbb{R}^n.$$

Supongamos $f \in \mathcal{L}^1(\mathbb{R}^n)$. Entonces se verifica

$$\int_{\mathbb{R}^n} \widehat{f}(y) e^{2\pi i \langle x, y \rangle} \Phi(\epsilon y) dy = (f * K_\epsilon)(x) \quad \forall x \in \mathbb{R}^n, \forall \epsilon \in \mathbb{R}^+.$$

Demostración. Comenzamos anotando los siguientes hechos:

- Por un lado, sabemos que la función $y \mapsto \widehat{f}(y) e^{2\pi i \langle x, y \rangle} \Phi(\epsilon y)$ es el producto de una función continua y acotada $y \mapsto \widehat{f}(y) e^{2\pi i \langle x, y \rangle}$ por una función integrable $y \mapsto \Phi(\epsilon y)$.
- Como K es la Transformada de Fourier de una función integrable en $\mathbb{R}^n$, sabemos que está acotada en $\mathbb{R}^n$. Luego también lo está K_ϵ , y eso hace que la convolución $f * K_\epsilon$ esté definida en todo $\mathbb{R}^n$.
- Como $\Phi(x) = \Phi(-x) \quad \forall x \in \mathbb{R}^n$, se sigue que $K(x) = K(-x)$, y entonces $K_\epsilon(x) = K_\epsilon(-x) \quad \forall x \in \mathbb{R}^n, \forall \epsilon \in \mathbb{R}^+$.

Fijamos $x \in \mathbb{R}^n$ y $\epsilon \in \mathbb{R}^+$. Se tiene que

$$\begin{aligned} \int_{\mathbb{R}^n} \widehat{f}(y) e^{2\pi i \langle x, y \rangle} \Phi(\epsilon y) dy &= \int_{\mathbb{R}^n} \widehat{\tau_{-x} f}(y) H_\epsilon \Phi(y) dy = \int_{\mathbb{R}^n} \tau_{-x} f(y) \widehat{H_\epsilon \Phi}(y) dy \\ &= \int_{\mathbb{R}^n} \tau_{-x} f(y) \epsilon^{-n} \widehat{\Phi}(\epsilon^{-1}y) dy = \int_{\mathbb{R}^n} \tau_{-x} f(y) K_\epsilon(y) dy \\ &= \int_{\mathbb{R}^n} f(x+y) K_\epsilon(y) dy = \int_{\mathbb{R}^n} f(x-y) K_\epsilon(-y) dy \\ &= \int_{\mathbb{R}^n} f(x-y) K_\epsilon(y) dy = (f * K_\epsilon)(x). \end{aligned}$$

Por tanto, hemos probado (11.3). □

Lema 5.7.2. *Sea $\Phi \in \mathcal{L}^1(\mathbb{R}^n)$ y supongamos que se cumplen las siguientes condiciones:*

- Φ es continua en 0,

- $\Phi(0) = 1$,
- $\widehat{\Phi} \in \mathcal{L}^1(\mathbb{R}^n)$.

Entonces,

$$\int_{\mathbb{R}^n} \widehat{\Phi}(y) dy = 1.$$

Demostración. Basta con aplicar el Teorema de Inversión a Φ , cuya Transformada es integrable en $\mathbb{R}^n$. Como Φ es continua en 0, se tiene que

$$1 = \Phi(0) = \int_{\mathbb{R}^n} \widehat{\Phi}(y) dy.$$

□

Teorema 5.7.3. Sea $\Phi \in \mathbb{R}^n$ y suponemos que se verifica que

- Φ es continua en 0,
- $\Phi(0) = 1$,
- $\widehat{\Phi} \in \mathcal{L}^1(\mathbb{R}^n)$.

Si $f \in \mathcal{L}^1(\mathbb{R}^n)$, entonces

$$\lim_{\substack{\epsilon \rightarrow 0 \\ \epsilon > 0}} \int_{\mathbb{R}^n} \left| f(x) - \int_{\mathbb{R}^n} \widehat{f}(y) e^{2\pi i \langle x, y \rangle} \Phi(\epsilon y) dy \right| dx = 0.$$

Demostración. Sea $K = \widehat{\Phi}$. Por un lado, sabemos por la proposición 5.7.1 que

$$\int_{\mathbb{R}^n} \widehat{f}(y) e^{2\pi i \langle x, y \rangle} \Phi(\epsilon y) dy = (f * K_\epsilon)(x) \quad \forall x \in \mathbb{R}^n, \forall \epsilon \in \mathbb{R}^+.$$

Por otra parte, sabemos que por el Lema 5.7.2 se cumple que

$$\int_{\mathbb{R}^n} K(x) dx = 1.$$

Por tanto, aplicando el Corolario 5.6.12 se tiene que

$$0 = \lim_{\substack{\epsilon \rightarrow 0 \\ \epsilon > 0}} \|f - f * K_\epsilon\|_1 = \lim_{\substack{\epsilon \rightarrow 0 \\ \epsilon > 0}} \int_{\mathbb{R}^n} \left| f(x) - \int_{\mathbb{R}^n} \widehat{f}(y) e^{2\pi i \langle x, y \rangle} \Phi(\epsilon y) dy \right| dx = 0. \quad \square$$

Teorema 5.7.4. Sea $\Phi \in \mathbb{R}^n$ y suponemos que se verifica que

- Φ es continua en 0,
- $\Phi(0) = 1$,
- $\widehat{\Phi} \in \mathcal{L}^\infty(\mathbb{R}^n)$.

Si $f \in \mathcal{L}^\infty(\mathbb{R}^n)$, entonces

- Supongamos que f es continua en cada punto de un subconjunto compacto $E \subset \mathbb{R}^n$.

Entonces

$$\lim_{\substack{\epsilon \rightarrow 0 \\ \epsilon > 0}} \int_{\mathbb{R}^n} \widehat{f}(y) e^{2\pi i \langle x, y \rangle} \Phi(\epsilon y) dy = f(x)$$

uniformemente en E , esto es,

$$\lim_{\substack{\epsilon \rightarrow 0 \\ \epsilon > 0}} \max_{x \in E} \left| f(x) - \int_{\mathbb{R}^n} \widehat{f}(y) e^{2\pi i \langle x, y \rangle} \Phi(\epsilon y) dy \right| = 0.$$

En particular, si f es continua en un punto $x \in \mathbb{R}^n$, entonces

$$\lim_{\substack{\epsilon \rightarrow 0 \\ \epsilon > 0}} \int_{\mathbb{R}^n} \widehat{f}(y) e^{2\pi i \langle x, y \rangle} \Phi(\epsilon y) dy = f(x).$$

- Si f es uniformemente continua en $\mathbb{R}^n$, entonces

$$\lim_{\substack{\epsilon \rightarrow 0 \\ \epsilon > 0}} \int_{\mathbb{R}^n} \widehat{f}(y) e^{2\pi i \langle x, y \rangle} \Phi(\epsilon y) dy = f(x)$$

uniformemente en $\mathbb{R}^n$, esto es,

$$\lim_{\substack{\epsilon \rightarrow 0 \\ \epsilon > 0}} \sup_{x \in \mathbb{R}^n} \left| f(x) - \int_{\mathbb{R}^n} \widehat{f}(y) e^{2\pi i \langle x, y \rangle} \Phi(\epsilon y) dy \right| = 0.$$

Demostración. Sea $K = \widehat{\Phi}$. Por un lado sabemos por la proposición 5.7.1 que

$$\int_{\mathbb{R}^n} \widehat{f}(y) e^{2\pi i \langle x, y \rangle} \Phi(\epsilon y) dy = (f * K_\epsilon)(x) \quad \forall x \in \mathbb{R}^n, \forall \epsilon \in \mathbb{R}^+.$$

Por otra parte, sabemos que por el Lema 5.7.2 se cumple que

$$\int_{\mathbb{R}^n} K(x) dx = 1.$$

Además,

$$|K(x)| = |\widehat{\Phi}(x)| \leq \|\Phi\|_1 \quad \forall x \in \mathbb{R}^n.$$

El Corolario 5.6.13 proporciona el resultado deseado. $\square$

Observación 5.7.5. Es posible prescindir de la condición $f \in \mathcal{L}^\infty(\mathbb{R}^n)$ con una teoría más extensa sobre los núcleos de sumabilidad. Este aspecto se reserva para trabajo futuro.

5.7.1. Método de Sumación de Gauss

Sea $f \in \mathcal{L}^1(\mathbb{R}^n)$. El método de sumación de Gauss consiste en utilizar el límite

$$\lim_{\substack{\epsilon \rightarrow 0 \\ \epsilon > 0}} \int_{\mathbb{R}^n} \widehat{f}(y) e^{2\pi i \langle x, y \rangle} e^{-\epsilon \|y\|^2} dy$$

como mecanismo de inversión de la Transformada de Fourier de f .

Para cada $\epsilon \in \mathbb{R}^+$ y cada $x \in \mathbb{R}^n$, se tiene que

$$\int_{\mathbb{R}^n} \widehat{f}(y) e^{2\pi i \langle x, y \rangle} e^{-\epsilon \|y\|^2} dy = \int_{\mathbb{R}^n} \widehat{f}(y) e^{2\pi i \langle x, y \rangle} G\left(\frac{\sqrt{\epsilon}}{\sqrt{\pi}} y\right) dy.$$

Sabemos que

- $G \in \mathcal{L}^1(\mathbb{R}^n)$,
- G es continua en $\mathbb{R}^n$,
- $G(0) = 1$,
- $G(x) = G(-x)$,
- $\widehat{G} = G \in \mathcal{L}^1(\mathbb{R}^n)$.

Corolario 5.7.6. Si $f \in \mathcal{L}^1(\mathbb{R}^n)$, entonces

$$\lim_{\substack{\epsilon \rightarrow 0 \\ \epsilon > 0}} \int_{\mathbb{R}^n} \left| f(x) - \int_{\mathbb{R}^n} \widehat{f}(y) e^{2\pi i \langle x, y \rangle} e^{-\epsilon \|y\|^2} dy \right| dx = 0.$$

Demostración. El resultado se deduce del Teorema 5.7.3, tomando $\Phi = G$ y como ϵ tomamos $\frac{\sqrt{\epsilon}}{\sqrt{\pi}}$. $\square$

Teorema 5.7.7. Sea $f \in \mathcal{L}^\infty(\mathbb{R}^n)$.

- Supongamos que f es continua en cada punto de un subconjunto compacto $E \subset \mathbb{R}^n$. Entonces

$$\lim_{\substack{\epsilon \rightarrow 0 \\ \epsilon > 0}} \int_{\mathbb{R}^n} \widehat{f}(y) e^{2\pi i \langle x, y \rangle} e^{-\epsilon \|y\|^2} dy = f(x)$$

uniformemente en E , esto es,

$$\lim_{\substack{\epsilon \rightarrow 0 \\ \epsilon > 0}} \max_{x \in E} \left| f(x) - \int_{\mathbb{R}^n} \widehat{f}(y) e^{2\pi i \langle x, y \rangle} e^{-\epsilon \|y\|^2} dy \right| = 0.$$

En particular, si f es continua en un punto $x \in \mathbb{R}^n$, entonces

$$\lim_{\substack{\epsilon \rightarrow 0 \\ \epsilon > 0}} \int_{\mathbb{R}^n} \widehat{f}(y) e^{2\pi i \langle x, y \rangle} e^{-\epsilon \|y\|^2} dy = f(x).$$

- Si f es uniformemente continua en $\mathbb{R}^n$, entonces

$$\lim_{\substack{\epsilon \rightarrow 0 \\ \epsilon > 0}} \int_{\mathbb{R}^n} \widehat{f}(y) e^{2\pi i \langle x, y \rangle} e^{-\epsilon \|y\|^2} dy = f(x)$$

uniformemente en $\mathbb{R}^n$, esto es,

$$\lim_{\substack{\epsilon \rightarrow 0 \\ \epsilon > 0}} \sup_{x \in \mathbb{R}^n} \left| f(x) - \int_{\mathbb{R}^n} \widehat{f}(y) e^{2\pi i \langle x, y \rangle} e^{-\epsilon \|y\|^2} dy \right| = 0.$$

Demostración. El resultado se deduce del Teorema 5.7.4, tomando $\Phi = G$ y como ϵ tomamos

CAPÍTULO 5. CONVOLUCIÓN

$$\frac{\sqrt{\epsilon}}{\sqrt{\pi}}.$$

□

Capítulo 6.

Transformada de Fourier en $\mathcal{L}^2(\mathbb{R}^n)$

Para motivar este capítulo comenzamos, haciendo alusión a las series de Fourier. El escenario fundamental en el que se desarrolla esta teoría era el espacio $\mathcal{L}^1(\mathbb{T})$ donde

$$\mathbb{T} = \{f \in \mathcal{L}^0(\mathbb{R}), \text{ } 2\pi\text{-periódicas e integrables en } [-\pi, \pi]\}.$$

En este caso conocíamos que $\mathcal{L}^2(\mathbb{T}) \subset \mathcal{L}^1(\mathbb{T})$, y por tanto $\mathcal{L}^2(\mathbb{T})$ heredaba toda la teoría desarrollada en $\mathcal{L}^1(\mathbb{T})$. De hecho, el comportamiento de las series era especialmente agradable en este espacio.

Motivados por este hecho, surge de manera natural la imperiosa necesidad de estudiar el comportamiento de la Transformada de Fourier en $\mathcal{L}^2(\mathbb{R}^n)$. Sin embargo, nuestras expectativas se verán frustradas en cuanto consideremos la proposición 3.1.2. Esta mostraba que los espacios $\mathcal{L}^p(\mathbb{R}^n)$ eran incomparables. En particular, $\mathcal{L}^2(\mathbb{R}^n) \not\subset \mathcal{L}^1(\mathbb{R}^n)$.

En efecto, la función $f : \mathbb{R}^n \rightarrow \mathbb{R}$ definida como:

$$f(x) = \begin{cases} x_1^{-1} \cdots x_n^{-1} & \text{si } x_i > 1 \ \forall i \in \{1, \dots, n\}, \\ 0 & \text{en otro caso,} \end{cases}$$

verifica que $f \in \mathcal{L}^2(\mathbb{R}^n)$, pero sin embargo $f \notin \mathcal{L}^1(\mathbb{R}^n)$.

Por tanto, el objeto $\hat{f}$ definido en 4.1.1 carece de sentido para f , y no podemos trasladar directamente los resultados teóricos estudiados en $\mathcal{L}^1(\mathbb{R}^n)$ a este espacio.

Los objetivos de este capítulo son, entonces, definir el concepto de la Transformada de Fourier $\hat{f}$ para cualquier función $f \in \mathcal{L}^2(\mathbb{R}^n)$, estudiar sus propiedades en este espacio, y analizar la relación que guarda con la definición previamente estudiada en $\mathcal{L}^1(\mathbb{R}^n)$. Para ello, usaremos como herramienta clave el Teorema de Plancharel que enunciaremos y demostramos a continuación.

6.1. Definición

Presentamos el Teorema de Plancharel, el cual pone de manifiesto que la Transformada de Fourier de una función $f \in \mathcal{L}^1(\mathbb{R}^n) \cap \mathcal{L}^2(\mathbb{R}^n)$ tiene un comportamiento especialmente agradable.

Lema 6.1.1. Sea $f \in \mathcal{L}^1(\mathbb{R}^n) \cap \mathcal{L}^2(\mathbb{R}^n)$. Entonces $\widehat{f} \in \mathcal{L}^2(\mathbb{R}^n)$ y se verifica que

$$\int_{\mathbb{R}^n} |\widehat{f}(y)|^2 dy = \int_{\mathbb{R}^n} |f(y)|^2 dy.$$

Equivalentemente,

$$\|f\|_{L_2} = \|\widehat{f}\|_{L_2}.$$

Demostración. Comenzamos observando los siguientes hechos:

- De la proposición 3.2.6 deducimos que $\widehat{\widehat{f}} \in \mathcal{L}^1(\mathbb{R}^n) \cap \mathcal{L}^2(\mathbb{R}^n)$. Además, por 4.2.2 sabemos que:

$$\widehat{\widehat{f}}(y) = \widehat{f}(y) \quad \forall y \in \mathbb{R}^n.$$

- Como $f, \widehat{f} \in \mathcal{L}^1(\mathbb{R}^n) \cap \mathcal{L}^2(\mathbb{R}^n)$, se tiene que la convolución

$$g = f * \widehat{\widehat{f}}$$

está definida en todo $\mathbb{R}^n$, y es uniformemente continua y acotada en $\mathbb{R}^n$. Nótese también que $g \in \mathcal{L}^1(\mathbb{R}^n)$, y por tanto podemos calcular su Transformada de Fourier

$$\widehat{g}(y) = \widehat{f}(y) \widehat{\widehat{f}(y)} = |\widehat{f}(y)|^2 \quad \forall y \in \mathbb{R}^n.$$

- Se verifica que

$$g(0) = (f * \widehat{\widehat{f}})(0) = \int_{\mathbb{R}^n} f(y) \widehat{\widehat{f}}(-y) dy = \int_{\mathbb{R}^n} f(y) \overline{\widehat{f}(y)} dy = \int_{\mathbb{R}^n} |f(y)|^2 dy. \quad (6.1)$$

Nuestras expectativas de poder aplicar el Teorema de Inversión a g se ven frustradas al no saber si $\widehat{g}$ está en $\mathcal{L}^1(\mathbb{R}^n)$. Es por ello por lo que recurrimos a nuestra teoría de núcleos de sumabilidad. Como g es una función uniformemente continua en $\mathbb{R}^n$, usando el método de sumación de Gauss, se tiene que:

$$g(x) = \lim_{\substack{\epsilon \rightarrow 0 \\ \epsilon > 0}} \int_{\mathbb{R}^n} \widehat{g}(y) e^{2\pi i \langle x, y \rangle} e^{-\epsilon \|y\|^2} dy \quad \forall x \in \mathbb{R}^n.$$

Concretamente, para $x = 0$ se cumple

$$g(0) = \lim_{\substack{\epsilon \rightarrow 0 \\ \epsilon > 0}} \int_{\mathbb{R}^n} \widehat{g}(y) e^{-\epsilon \|y\|^2} dy \quad \forall x \in \mathbb{R}^n.$$

En particular, si tomamos una sucesión concreta que tiende a 0:

$$g(0) = \lim_{m \rightarrow \infty} \int_{\mathbb{R}^n} |\widehat{f}(y)|^2 e^{-\frac{1}{m} \|y\|^2} dy \quad \forall x \in \mathbb{R}^n. \quad (6.2)$$

Por otro lado, aplicamos el teorema de la convergencia monótona para funciones medibles positivas a la sucesión creciente

$$\{|\widehat{f}(y)|^2 e^{-\frac{1}{m} \|y\|^2}\},$$

obteniendo

$$\lim_{m \rightarrow \infty} \int_{\mathbb{R}^n} |\widehat{f}(y)|^2 e^{-\frac{1}{m} \|y\|^2} dy = \int_{\mathbb{R}^n} \lim_{m \rightarrow \infty} |\widehat{f}(y)|^2 e^{-\frac{1}{m} \|y\|^2} dy = \int_{\mathbb{R}^n} |\widehat{f}(y)|^2 dy \quad \forall y \in \mathbb{R}^n.$$

Usando (6.1), deducimos

$$g(0) = \int_{\mathbb{R}^n} |\widehat{f}(y)|^2 dy.$$

Luego por (6.1) concluimos que $\widehat{f} \in \mathcal{L}^2(\mathbb{R}^n)$ y además

$$\int_{\mathbb{R}^n} |\widehat{f}(y)|^2 dy = \int_{\mathbb{R}^n} |f(y)|^2 dy. \quad \square$$

Antes de introducir el concepto de Transformada de Fourier en $\mathcal{L}^2(\mathbb{R}^n)$, recalamos un hecho que jugará un papel importante a lo largo de este capítulo.

Proposición 6.1.2. *El espacio $\mathcal{L}^1(\mathbb{R}^n) \cap \mathcal{L}^2(\mathbb{R}^n)$ es un subespacio vectorial denso en $\mathcal{L}^2(\mathbb{R}^n)$.*

Demostración. La única consideración que debemos tener presente es que por el Teorema 3.1.7, C_{00} es un subespacio vectorial denso en $\mathcal{L}^2(\mathbb{R}^n)$. Y además, $\mathcal{L}^1(\mathbb{R}^n) \cap \mathcal{L}^2(\mathbb{R}^n)$ contiene a $C_{00}(\mathbb{R}^n)$, por lo que se sigue que $\mathcal{L}^1(\mathbb{R}^n) \cap \mathcal{L}^2(\mathbb{R}^n)$ es un subespacio vectorial denso en $\mathcal{L}^2(\mathbb{R}^n)$. $\square$

Teorema 6.1.3 (Plancharel). *Existe un único operador lineal y continuo*

$$\mathcal{F} : L^2(\mathbb{R}^n) \rightarrow L^2(\mathbb{R}^n)$$

tal que

$$\mathcal{F}(f) = \widehat{f} \quad \forall f \in L^1(\mathbb{R}^n) \cap L^2(\mathbb{R}^n). \quad (6.3)$$

Además, el operador $\mathcal{F}$ es biyectivo y cumple la identidad

$$\int_{\mathbb{R}^n} |\mathcal{F}(f)(y)|^2 dy = \int_{\mathbb{R}^n} |f(y)|^2 dy.$$

Demostración. Notemos los siguientes puntos

- El lema 6.1.1 pone de manifiesto que la Transformada de Fourier define un operador lineal y continuo de $L^1(\mathbb{R}^n) \cap L^2(\mathbb{R}^n)$ en $L^2(\mathbb{R}^n)$.
- Por la proposición 6.1.2, sabemos que el espacio $\mathcal{L}^1(\mathbb{R}^n) \cap \mathcal{L}^2(\mathbb{R}^n)$ es un subespacio vectorial denso en $\mathcal{L}^2(\mathbb{R}^n)$.

El teorema de extensión de operadores en espacios de Banach afirma que existe un único operador lineal y continuo $\mathcal{F} : L^2(\mathbb{R}^n) \rightarrow L^2(\mathbb{R}^n)$ que extiende la Transformada de Fourier en $L^1(\mathbb{R}^n) \cap L^2(\mathbb{R}^n)$, de modo que

$$\mathcal{F}(f) = \widehat{f} \quad \forall f \in L^1(\mathbb{R}^n) \cap L^2(\mathbb{R}^n).$$

Se prueba así la primera parte del enunciado. Proseguimos demostrando la identidad (6.1.3).

CAPÍTULO 6. TRANSFORMADA DE FOURIER EN $\mathcal{L}^2(\mathbb{R}^n)$

Para ello, usaremos como es natural el lema 6.1.1, que prueba la identidad para funciones en $L^1(\mathbb{R}^n) \cap L^2(\mathbb{R}^n)$.

Sea $f \in L^2(\mathbb{R}^n)$. Tomamos una sucesión $\{f_m\} \in L^1(\mathbb{R}^n) \cap L^2(\mathbb{R}^n)$ tal que

$$\lim_{m \rightarrow \infty} \|f_m - f\|_2 = 0.$$

Se tiene entonces

$$\lim_{m \rightarrow \infty} \|\mathcal{F}(f_m) - \mathcal{F}(f)\|_2 = 0.$$

Además, sabemos que

$$\|\mathcal{F}(f_m)\|_2 = \|\widehat{f_m}\|_2 = \|f_m\|_2,$$

de donde deducimos lo siguiente

$$\|\mathcal{F}(f)\|_2 = \lim_{m \rightarrow \infty} \|\mathcal{F}(f_m)\|_2 = \lim_{m \rightarrow \infty} \|f_m\|_2 = \|f\|_2,$$

probando así la identidad (6.1.3). A continuación, probaremos la última aseveración del teorema. Para ello, tenemos que comprobar que el operador $\mathcal{F}$ es inyectivo y sobreyectivo.

- Es claro que $\mathcal{F}$ es inyectivo, ya que si $f \in \text{Ker}(\mathcal{F})$, entonces

$$0 = \|\mathcal{F}(f)\|_2 = \|f\|_2,$$

luego necesariamente $f = 0$.

- Veamos que $\mathcal{F}$ es sobreyectivo. Para ello probaremos que $\mathcal{F}(L^2(\mathbb{R}^n)) = L^2(\mathbb{R}^n)$.

Sea $f \in \mathcal{L}^1(\mathbb{R}^n) \cap \mathcal{L}^2(\mathbb{R}^n)$. Definimos para cada $m \in \mathbb{N}$, la sucesión de funciones $\{f_m\} : \mathbb{R} \rightarrow \mathbb{C}$ dada por

$$f_m(x) = \widehat{f}(-x)e^{-\frac{1}{m}\|x\|^2} \quad \forall x \in \mathbb{R}^n.$$

Sabemos que para cada $m \in \mathbb{N}$ se tiene que $f_m \in \mathcal{L}^1(\mathbb{R}^n) \cap \mathcal{L}^2(\mathbb{R}^n)$, ya que en efecto al estar $f \in \mathcal{L}^1(\mathbb{R}^n)$, $\widehat{f}$ está acotada y el factor $e^{-\frac{1}{m}\|x\|^2} \in \mathcal{L}^1(\mathbb{R}^n) \cap \mathcal{L}^2(\mathbb{R}^n)$.

Nótese por la proposición 5.6.5 que

$$\begin{aligned} \widehat{f_m}(y) &= \int_{\mathbb{R}^n} \widehat{f}(-x)e^{-\frac{1}{m}\|x\|^2} e^{-2\pi i \langle x, y \rangle} dx = \int_{\mathbb{R}^n} \widehat{f}(x)e^{-\frac{1}{m}\|x\|^2} e^{2\pi i \langle x, y \rangle} dx \\ &= (f * K_m)(y) \quad \forall y \in \mathbb{R}, \forall m \in \mathbb{N}. \end{aligned}$$

con $K_m(x) = (\sqrt{\pi m})^n G(\sqrt{\pi m}x) \quad \forall x \in \mathbb{R}, \forall n \in \mathbb{N}$. Luego por el Teorema 5.6.11, se tiene que

$$\lim_{m \rightarrow \infty} \|\mathcal{F}(f_m) - f\|_2 = \lim_{m \rightarrow \infty} \|\widehat{f_m} - f\|_2 = \lim_{m \rightarrow \infty} \|f * K_m - f\|_2 = 0.$$

Por tanto, se tiene que $f \in \overline{\mathcal{F}(L^2(\mathbb{R}^n))}$, y como consecuencia $L^1(\mathbb{R}^n) \cap L^2(\mathbb{R}^n) \subset \overline{\mathcal{F}(L^2(\mathbb{R}^n))}$. Finalmente,

$$L^2(\mathbb{R}^n) = \overline{L^1(\mathbb{R}^n) \cap L^2(\mathbb{R}^n)} \subset \overline{\mathcal{F}(L^2(\mathbb{R}^n))}.$$

donde hemos usado que $\mathcal{F}$ es una isometría de $L^2(\mathbb{R}^n)$ en $\mathcal{F}(L^2(\mathbb{R}^n))$, pudiendo así asegurar que el espacio $\mathcal{F}(L^2(\mathbb{R}^n))$ es un subespacio completo de $L^2(\mathbb{R}^n)$, y entonces cerrado en

6.2. PERMUTACIÓN INTEGRATORIA DE LA TRANSFORMADA

este, llegando a $(L^2(\mathbb{R}^n)) = \mathcal{F}(L^2(\mathbb{R}^n))$. Se tiene lo que queríamos, $\mathcal{F}(L^2(\mathbb{R}^n)) = L^2(\mathbb{R}^n)$. Por tanto, $\mathcal{F}$ es un operador biyectivo de $L^2(\mathbb{R}^n)$ en $L^2(\mathbb{R}^n)$. $\square$

Observación 6.1.4. La identidad $\mathcal{F}(f) = \widehat{f}$ para toda función $f \in L^1(\mathbb{R}) \cap L^2(\mathbb{R})$ se debe entender en el sentido de que la clase de equivalencia de la función $\widehat{f}$ es la misma que la de $\mathcal{F}(f)$. Esto significa que $\mathcal{F}(f)(y) = f(y)$ para casi todo $y \in \mathbb{R}^n$.

Observación 6.1.5. Para cada $f \in L^2(\mathbb{R}^n)$, utilizaremos la notación $\widehat{f}$ para referirnos a $\mathcal{F}(f)$. Por tanto, $\widehat{f}$ se considerará una clase de equivalencia de $L^2(\mathbb{R}^n)$, aunque habitualmente se piensa en $\widehat{f}$ simplemente como una función, eligiendo un representante de su clase. En el caso particular de que $f \in L^1(\mathbb{R}^n) \cap L^2(\mathbb{R}^n)$, entonces $\widehat{f}$ será interpretada de manera convencional como una función en lugar de como una clase de equivalencia.

Como exploraremos en secciones futuras, hay una similitud considerable en las propiedades de la Transformada de Fourier en $\mathcal{L}^1(\mathbb{R}^n)$ y $\mathcal{L}^2(\mathbb{R}^n)$. A pesar de esto, existen algunas diferencias destacables entre ambas.

Dada una función $f \in \mathcal{L}^1(\mathbb{R}^n)$,

- La clase $\widehat{f}$ puede carecer de una función representante continua;
- La clase $\widehat{f}$ puede carecer de una función representante acotada;
- La clase $\widehat{f}$ puede carecer de una función representante que cumple el lema de Riemann-Lebesgue.

Ejemplo 6.1.6. Sea $f : \mathbb{R}^n \rightarrow \mathbb{C}$,

$$f(x) = \sum_{m=1}^{\infty} m \mathcal{X}_{]m, m + \frac{1}{2^m}[^n}(x).$$

Se puede comprobar que $f \in \mathcal{L}^2(\mathbb{R}^n)$, por lo que es la Transformada de Fourier de alguna función en $\mathcal{L}^2(\mathbb{R}^n)$. Sin embargo, es sencillo comprobar que f tiene discontinuidades de salto, por lo que no puede coincidir casi por doquier con una función continua.

6.2. Permutación Integratoria de la Transformada

Esta propiedad era ya conocida en el espacio $\mathcal{L}^1(\mathbb{R}^n)$. Ahora examinemos cómo el resultado es análogo en $\mathcal{L}^2(\mathbb{R}^n)$.

Teorema 6.2.1. Sean $f, g \in \mathcal{L}^2(\mathbb{R}^n)$. Entonces,

$$\int_{\mathbb{R}^n} \widehat{f}(x) g(x) dx = \int_{\mathbb{R}^n} f(x) \widehat{g}(x) dx. \quad (6.4)$$

Demostración.

Notemos que el producto de dos funciones en $\mathcal{L}^2(\mathbb{R}^n)$ es integrable en $\mathbb{R}^n$, lo que justifica las dos integrales presentes en la ecuación (6.2.1).

Ahora estimaremos la siguiente cantidad.

$$\left| \int_{\mathbb{R}^n} \widehat{f}(x)g(x) dx - \int_{\mathbb{R}^n} f(x)\widehat{g}(x) dx \right|.$$

Tomamos dos sucesiones $\{f_m\}, \{g_m\}$ en $\mathcal{L}^1(\mathbb{R}^n) \cap \mathcal{L}^2(\mathbb{R}^n)$ tales que

$$\lim_{m \rightarrow \infty} \|f - f_m\|_2 = \lim_{m \rightarrow \infty} \|g - g_m\|_2 = 0.$$

Como la Transformada de Fourier de una función en $\mathcal{L}^1(\mathbb{R}^n)$ es uniformemente continua en $\mathbb{R}^n$, se sigue que

$$\lim_{m \rightarrow \infty} \|\widehat{f} - \widehat{f}_m\|_2 = \lim_{m \rightarrow \infty} \|\widehat{g} - \widehat{g}_m\|_2 = 0.$$

Además, también ocurre

$$\lim_{m \rightarrow \infty} \|\widehat{f}_m\|_2 = \lim_{m \rightarrow \infty} \|f_m\|_2 = \|f\|_2, \quad \lim_{m \rightarrow \infty} \|\widehat{g}_m\|_2 = \lim_{m \rightarrow \infty} \|g_m\|_2 = \|g\|_2.$$

Entonces, dado $m \in \mathbb{N}$, se verifica la siguiente desigualdad

$$\begin{aligned} & \left| \int_{\mathbb{R}^n} \widehat{f}(x)g(x) dx - \int_{\mathbb{R}^n} f(x)\widehat{g}(x) dx \right| \\ &= \left| \int_{\mathbb{R}^n} [\widehat{f}(x)g(x) - \widehat{f}_m(x)g_m(x) + \widehat{f}_m(x)g_m(x) - f(x)\widehat{g}(x)] dx \right| \\ &\leq \int_{\mathbb{R}^n} |\widehat{f}(x)g(x) - \widehat{f}_m(x)g_m(x)| dx + \int_{\mathbb{R}^n} |f_m(x)\widehat{g}_m(x) - \widehat{f}(x)\widehat{g}(x)| dx \\ &\leq \int_{\mathbb{R}^n} |(\widehat{f} - \widehat{f}_m)(x)g(x)| dx + \int_{\mathbb{R}^n} |\widehat{f}_m(x)(g - g_m)(x)| dx \\ &+ \int_{\mathbb{R}^n} |(f_m - f)(x)\widehat{g}_m(x)| dx + \int_{\mathbb{R}^n} |\widehat{f}(x)(\widehat{g}_m - \widehat{g})(x)| dx \\ &\leq \|\widehat{f} - \widehat{f}_m\|_2 \|g\|_2 + \|\widehat{f}_m\|_2 \|g - g_m\|_2 + \|f_m - f\|_2 \|g_m\|_2 + \|f\|_2 \|\widehat{g}_m - \widehat{g}\|_2. \end{aligned}$$

Luego tomando límite concluimos que,

$$\left| \int_{\mathbb{R}^n} \widehat{f}(x)g(x) dx - \int_{\mathbb{R}^n} f(x)\widehat{g}(x) dx \right| = 0,$$

de donde se deduce la fórmula mencionada. □

6.3. Propiedades

A continuación, examinaremos algunas propiedades análogas a las estudiadas previamente en $\mathcal{L}^1(\mathbb{R}^n)$. De hecho, su demostración se deriva directamente de éstas.

Proposición 6.3.1. Sean $f, g \in \mathcal{L}^1(\mathbb{R}^n)$. Entonces

1. $\widehat{\widehat{f}} = \widehat{\widetilde{f}}$,
2. $\widetilde{\widehat{f}} = \widetilde{\widetilde{f}}$,
3. $\widehat{\widetilde{f}} = \widetilde{\widehat{f}}$.

Demostración. Tomamos una sucesión $\{f_m\}$ en $\mathcal{L}^1(\mathbb{R}^n) \cap \mathcal{L}^2(\mathbb{R}^n)$ tal que

$$\lim_{m \rightarrow \infty} \|f - f_m\|_2 = 0.$$

Por la continuidad de las operaciones de conjugación $f \mapsto \overline{f}$ y simetría $f \mapsto \widetilde{f}$, y usando que la Transformada de Fourier $f \mapsto \mathcal{F}(f)$ es un operador continuo en $L^2(\mathbb{R}^n)$,

$$\lim_{m \rightarrow \infty} \|\widehat{f} - \widehat{f_m}\|_2 = \|\widehat{\widehat{f_m}} - \widehat{\widetilde{f}}\|_2 = 0.$$

Como para cada $m \in \mathbb{N}$, $f_m \in \mathcal{L}^1(\mathbb{R}^n)$, podemos entonces aplicar el resultado de la Proposición 4.2.2:

$$\|\widehat{f} - \widetilde{\widehat{f}}\|_2 = \|\widehat{f} - \widehat{f_m} - \widehat{f_m} + \widetilde{\widehat{f_m}}\|_2 \leq \|\widehat{f} - \widehat{f_m}\|_2 + \|\widehat{\widehat{f_m}} - \widetilde{\widehat{f}}\|_2.$$

Tomando límite se tiene el primer apartado de la proposición.

El resto de igualdades se demuestran de manera análoga, ya que lo que se ha usado realmente es la continuidad de todos los operadores involucrados y la densidad del espacio $L^1(\mathbb{R}^n) \cap L^2(\mathbb{R}^n)$ en $L^2(\mathbb{R}^n)$, junto con la validez de las fórmulas presentadas para cualquier función de $L^1(\mathbb{R}^n) \cap L^2(\mathbb{R}^n)$. □

Proposición 6.3.2. Sea $f \in \mathcal{L}^2(\mathbb{R}^n)$ y sea $a \in \mathbb{R}^+$. Entonces, se tiene:

$$\widehat{H_a f} = a^{-n} H_{a^{-1}} \widehat{f}.$$

Demostración. Tomamos una sucesión $\{f_m\}$ en $\mathcal{L}^1(\mathbb{R}^n) \cap \mathcal{L}^2(\mathbb{R}^n)$ tal que

$$\lim_{m \rightarrow \infty} \|f - f_m\|_2 = 0.$$

Por la continuidad de las operaciones de conjugación $f \mapsto H_a f$, y usando que $\widehat{\cdot}$ es un operador continuo en $L^2(\mathbb{R}^n)$,

$$\lim_{m \rightarrow \infty} \|\widehat{H_a f} - \widehat{H_a f_m}\|_2 = \lim_{m \rightarrow \infty} \|a^{-n} H_{a^{-1}} \widehat{f_m} - a^{-n} H_{a^{-1}} \widehat{f}\|_2 = 0.$$

Como para cada $m \in \mathbb{N}$, $f_m \in \mathcal{L}^1(\mathbb{R}^n)$, podemos entonces aplicar el resultado de la Proposición 4.2.3:

$$\|\widehat{H_a f} - a^{-n} H_{a^{-1}} \widehat{f}\|_2 = \|\widehat{H_a f} - \widehat{H_a f_m} + \widehat{H_a f_m} - a^{-n} H_{a^{-1}} \widehat{f}\|_2$$

$$\leq \|\widehat{H_a f} - \widehat{H_a f_m}\|_2 + \|a^{-n} H_{a^{-1}} \widehat{f_m} - a^{-n} H_{a^{-1}} \widehat{f}\|_2.$$

Tomando límite se concluye el resultado. $\square$

Proposición 6.3.3. Sea $f \in \mathcal{L}^2(\mathbb{R}^n)$ y sea $t \in \mathbb{R}^n$. Entonces se tiene

$$\widehat{\tau_t f} = \mu_{-t} \widehat{f}.$$

Demostración. Repetimos el argumento usado en las proposiciones anteriores. La continuidad de los operadores involucrados y la densidad del espacio $L^1(\mathbb{R}^n) \cap L^2(\mathbb{R}^n)$ en $L^2(\mathbb{R}^n)$ junto con la validez de las fórmulas presentadas para cualquier función de $L^1(\mathbb{R}^n) \cap L^2(\mathbb{R}^n)$ proporciona el resultado para funciones de $L^2(\mathbb{R}^n)$. $\square$

Proposición 6.3.4. Sea $f \in \mathcal{L}^2(\mathbb{R}^n)$ y $t \in \mathbb{R}^n$. Entonces, se tiene:

$$\widehat{\mu_t f} = \tau_t \widehat{f}.$$

Demostración. Repetimos el argumento usado en las proposiciones anteriores. La continuidad de los operadores involucrados y la densidad del espacio $L^1(\mathbb{R}^n) \cap L^2(\mathbb{R}^n)$ en $L^2(\mathbb{R}^n)$ junto con la validez de las fórmulas presentadas para cualquier función de $L^1(\mathbb{R}^n) \cap L^2(\mathbb{R}^n)$ proporciona el resultado para funciones de $L^2(\mathbb{R}^n)$. $\square$

6.4. Fórmulas de Parseval

Proposición 6.4.1. Sean $f, g \in \mathcal{L}^2(\mathbb{R}^n)$. Entonces se cumple la identidad

$$\int_{\mathbb{R}^n} f(x) \overline{g(x)} dx = \int_{\mathbb{R}^n} \widehat{f}(x) \overline{\widehat{g}(x)} dx. \quad (6.5)$$

Demostración. Notemos que el producto de dos funciones en $\mathcal{L}^2(\mathbb{R}^n)$ es integrable en $\mathbb{R}^n$, lo que justifica las dos integrales presentes en la ecuación (6.4.1). Se tiene que

$$\int_{\mathbb{R}^n} \widehat{f}(x) \overline{\widehat{g}(x)} dx = \langle \widehat{f}, \widehat{g} \rangle. \quad (6.6)$$

Como $f, g \in \mathcal{L}^2(\mathbb{R}^n)$, podemos usar la identidad de polarización para la prueba. En efecto,

$$\begin{aligned} \langle \widehat{f}, \widehat{g} \rangle &= \frac{1}{4} \left[\|\widehat{f} + \widehat{g}\|_2^2 - \|\widehat{f} - \widehat{g}\|_2^2 + i\|\widehat{f} + i\widehat{g}\|_2^2 - i\|\widehat{f} - i\widehat{g}\|_2^2 \right] \\ &= \frac{1}{4} \left[\|\widehat{f + g}\|_2^2 - \|\widehat{f - g}\|_2^2 + i\|\widehat{f + ig}\|_2^2 - i\|\widehat{f - ig}\|_2^2 \right] \\ &= \frac{1}{4} \left[\|f + g\|_2^2 - \|f - g\|_2^2 + i\|f + ig\|_2^2 - i\|f - ig\|_2^2 \right] \\ &= \langle f, g \rangle. \end{aligned}$$

Notemos que

$$\int_{\mathbb{R}^n} f(x) \overline{g(x)} dx = \langle f, g \rangle. \quad (6.7)$$

Teniendo en cuenta (6.4) y (6.4), se concluye la prueba. $\square$

Tomando $f = g$ se obtiene

$$\int_{\mathbb{R}^n} |\widehat{f}(y)|^2 dy = \int_{\mathbb{R}^n} |f(y)|^2 dy.$$

Esta igualdad, previamente establecida en 6.1.1, es usada en el campo de teoría de señales. Es común encontrar una variante de esta fórmula ajustada por un factor que depende de 2π , debido a una elección distinta en la definición de la Transformada de Fourier. La fórmula muestra cómo la energía total de una señal f es equivalente a la energía total de su Transformada de Fourier $\widehat{f}$ distribuida a través de todas sus componentes frecuenciales. Este principio, fundamental en el análisis de señales, subraya la capacidad de la Transformada de Fourier para preservar la energía de una señal a medida que se traslada del dominio del tiempo al dominio de la frecuencia.

Corolario 6.4.2. Sean $f, g \in \mathcal{L}^2(\mathbb{R}^n)$. Entonces se cumple la identidad

$$\int_{\mathbb{R}^n} f(x)g(x) dx = \int_{\mathbb{R}^n} \widehat{f}(x)\widehat{g}(-x) dx.$$

Demostración. Usando la proposición 6.4.1 se tiene

$$\int_{\mathbb{R}^n} f(x)g(x) dx = \int_{\mathbb{R}^n} f(x)\overline{\widehat{g}(x)} dx = \int_{\mathbb{R}^n} \widehat{f}(x)\overline{\widehat{g}(x)} dx.$$

Finalmente, haciendo uso del primer apartado de la proposición, 6.3.1

$$\int_{\mathbb{R}^n} \widehat{f}(x)\overline{\widehat{g}(x)} dx = \int_{\mathbb{R}^n} \widehat{f}(x)\overline{\widehat{g}(-x)} dx = \int_{\mathbb{R}^n} \widehat{f}(x)\widehat{g}(-x) dx,$$

concluyendo lo que se pedía. $\square$

6.5. Convolución

En este contexto teórico, introducimos el Teorema de Convolución, cuya conclusión coincide con el que ya hemos demostrado previamente para $\mathcal{L}^1(\mathbb{R}^n)$.

Teorema 6.5.1. (*Teorema de Convolución*). Sean $f \in \mathcal{L}^1(\mathbb{R}^n), g \in \mathcal{L}^2(\mathbb{R}^n)$. Entonces se tiene que

$$\widehat{f * g} = \widehat{f}\widehat{g}.$$

Demostración. Naturalmente, nuestro objetivo es aplicar el resultado obtenido para funciones en $\mathcal{L}^1(\mathbb{R}^n)$. En efecto, $f \in \mathcal{L}^1(\mathbb{R}^n)$, sin embargo, $g \notin \mathcal{L}^1(\mathbb{R}^n)$, pero esto no es problema ya que podemos razonar como hemos hecho anteriormente. Tomamos una sucesión de funciones $\{g_m\}$

CAPÍTULO 6. TRANSFORMADA DE FOURIER EN $\mathcal{L}^2(\mathbb{R}^n)$

en $\mathcal{L}^1(\mathbb{R}^n) \cap \mathcal{L}^2(\mathbb{R}^n)$ de modo que

$$\lim_{m \rightarrow \infty} \|g - g_m\|_2 = 0.$$

Como $f \in \mathcal{L}^1(\mathbb{R}^n)$ y para cada $m \in \mathbb{N}$, $g_m \in \mathcal{L}^1(\mathbb{R}^n)$, podemos entonces aplicar el Teorema 5.2.1, obteniendo que

$$\widehat{f * g_m} = \widehat{f} \widehat{g_m} \quad \forall m \in \mathbb{N}.$$

En consecuencia, para cada $m \in \mathbb{N}$,

$$\begin{aligned} \|\widehat{fg} - \widehat{f} \widehat{g}\|_2 &= \|\widehat{fg} - \widehat{fg_m} + \widehat{fg_m} - \widehat{f} \widehat{g}\|_2 \\ &\leq \|\widehat{fg} - \widehat{fg_m}\|_2 + \|\widehat{fg_m} - \widehat{f} \widehat{g}\|_2 \\ &= \|(f * (g - g_m))^\wedge\|_2 + \|\widehat{f}(\widehat{g_m} - \widehat{g})\|_2 \\ &\leq \|(f * (g - g_m))\|_2 + \|\widehat{f}\|_\infty \|g - g_m\|_2 \leq 2\|f\|_1 \|g - g_m\|_2. \end{aligned}$$

Tomando límite se tiene que

$$\|\widehat{fg} - \widehat{f} \widehat{g}\|_2 \leq \lim_{m \rightarrow \infty} 2\|f\|_1 \|g - g_m\|_2 = 0,$$

lo cual demuestra $\widehat{f * g} = \widehat{f} \widehat{g}$. □

Observación 6.5.2. El lector podría sorprenderse al observar que, a diferencia de la mayoría de los enunciados presentados en este capítulo, en este caso las dos funciones no pertenecen a $\mathcal{L}^2(\mathbb{R}^n)$. Esto se debe a que en el caso de que ambas funciones estén en $\mathcal{L}^2(\mathbb{R}^n)$, la convolución de estas no tiene por qué estar en $\mathcal{L}^2(\mathbb{R}^n)$ necesariamente. Es por ello que imponemos que una esté en $\mathcal{L}^1(\mathbb{R}^n)$, en cuyo caso sí se tiene por el Teorema 5.4.3.

Como ya adelantábamos en capítulos anteriores, el siguiente resultado no tenía sentido en $\mathcal{L}^1(\mathbb{R}^n)$. Sin embargo, el producto de funciones en $\mathcal{L}^2(\mathbb{R}^n)$ es una función integrable en $\mathbb{R}^n$, lo cual permite presentar el siguiente resultado.

Teorema 6.5.3. Sean $f, g \in \mathcal{L}^2(\mathbb{R}^n)$. Entonces se tiene que

$$\widehat{fg} = \widehat{f} * \widehat{g}.$$

Demostración. Dado $y \in \mathbb{R}^n$, usando el Corolario 6.4.2, se obtiene que

$$\begin{aligned} (\widehat{fg})(y) &= \int_{\mathbb{R}^n} f(x)g(x)e^{-2\pi i \langle x, y \rangle} dx = \int_{\mathbb{R}^n} f(x)(\mu_{-y}g)(x) dx = \int_{\mathbb{R}^n} \widehat{f}(t)(\widehat{\mu_{-y}g})(-t) dt \\ &= \int_{\mathbb{R}^n} \widehat{f}(t)\widehat{g}(y - t) dt = (\widehat{f} * \widehat{g})(y). \end{aligned} \quad \square$$

De este modo, la convolución en el dominio temporal se convierte en multiplicación en el dominio de la frecuencia, mientras que la operación de multiplicar en el dominio temporal se traduce en realizar la convolución en el dominio de la frecuencia.

Una posible aplicación de este último resultado en el ámbito computacional es el diseño de filtros complejos a partir de la combinación de filtros más simples. Al multiplicar dos señales en el tiempo y tomar la Transformada de Fourier del producto, se puede obtener el efecto equivalente a convolucionar sus espectros, lo que es útil para la creación de respuestas de filtro específicas.

6.6. Teorema de Inversión

Probaremos a continuación el Teorema de Inversión. Como ya estudiamos en capítulos anteriores, este proporciona una manera de recuperar la función a partir de la Transformada de Fourier de esta. Sin embargo, en este marco teórico no hacen falta hipótesis adicionales como ocurría en $\mathcal{L}^1(\mathbb{R}^n)$.

Teorema 6.6.1. *Sea $f \in \mathcal{L}^2(\mathbb{R}^n)$. Entonces*

$$f(x) = \widehat{\widehat{f}}(-x) \text{ c.p.d en } \mathbb{R}^n.$$

Demostración. Dada $g \in \mathcal{L}^2(\mathbb{R}^n)$. Usando la proposición 6.4.1 se tiene

$$\int_{\mathbb{R}^n} \widetilde{f}(x) \overline{g(x)} dx = \int_{\mathbb{R}^n} \widehat{\widehat{f}}(x) \overline{\widehat{g}(x)} dx = \int_{\mathbb{R}^n} \widehat{\widehat{f}}(-x) \overline{\widehat{g}(x)} dx = \int_{\mathbb{R}^n} \widehat{\widehat{f}}(x) \overline{\widehat{g}(-x)} dx.$$

Por el Corolario 6.4.2, sabemos que

$$\int_{\mathbb{R}^n} \widehat{\widehat{f}}(x) \overline{\widehat{g}(-x)} dx = \int_{\mathbb{R}^n} \widehat{\widehat{f}}(x) \overline{\widehat{g}(x)} dx.$$

Luego,

$$\int_{\mathbb{R}^n} \widetilde{f}(x) \overline{g(x)} dx = \int_{\mathbb{R}^n} \widehat{\widehat{f}}(x) \overline{\widehat{g}(x)} dx \quad \forall g \in \mathcal{L}^2(\mathbb{R}^n).$$

De donde deducimos que $\widetilde{f} = \widehat{\widehat{f}}$, y por tanto $f(x) = \widehat{\widehat{f}}(-x)$ c.p.d en $\mathbb{R}^n$. □

Parte II.

Parte Informática. Aceleración de Redes Neuronales Convolucionales mediante Análisis en el Dominio de la Frecuencia

Capítulo 7.

Introducción a la Parte Informática

El DL [7] ha experimentado un notable crecimiento en la última década, convirtiéndose en una de las áreas más dinámicas y en expansión dentro del campo de la IA. Este avance se caracteriza por la utilización de redes neuronales profundas que permiten a los modelos aprender múltiples niveles de representación que corresponden con diferentes niveles de abstracción. El crecimiento en este campo ha impulsado significativamente otras áreas dentro de la comunidad científica, como por ejemplo el área de VC [2, 3, 4] donde las CNN [7] son especialmente importantes debido a su habilidad para procesar y analizar imágenes con una eficacia que supera ampliamente a los métodos tradicionales.

Lo cierto es que a medida que se busca resolver tareas cada vez más desafiantes, la necesidad de modelos más complejos se hace evidente. Esto implica la utilización de volúmenes de datos mucho mayores para potenciar estos modelos avanzados y evitar el sobreajuste. Este fenómeno se conoce como “La maldición de la dimensionalidad”, término acuñado por el matemático Richard Bellman en 1957 [25]. La maldición de la dimensionalidad hace referencia a algunos problemas que surgen cuando se trabaja con datos en espacios de alta dimensionalidad. Esto se debe a que en espacios de alta dimensionalidad, los puntos de datos tienden a estar mucho más alejados entre sí, lo que dificulta la detección de patrones o clusters significativos. Esto resulta en modelos complejos que pueden terminar aprendiendo peculiaridades y ruidos específicos de los datos de entrenamiento en lugar de identificar patrones generales. Este problema, conocido como sobreajuste [26], se agrava si no se cuenta con una cantidad suficiente de datos que cubra adecuadamente el espacio de alta dimensionalidad, impidiendo que el modelo generalice bien a datos nuevos. Por ello, para mitigar los problemas de sobreajuste en contextos de alta dimensionalidad, una de las estrategias más efectivas es aumentar la cantidad de datos disponibles de manera que el modelo tenga más ejemplos para aprender y pueda desarrollar una representación más robusta y generalizable del problema. Como consecuencia, el incremento de la dimensionalidad del problema conlleva un necesario aumento de los datos para el entrenamiento del modelo.

Por lo que, mientras que los primeros conjuntos de datos en AA [8, 9] incluían miles o decenas de miles de muestras [12, 13, 14], los actuales comprenden millones [15, 16]. Este incremento plantea desafíos significativos para entrenar modelos de manera eficiente en tiempos razonables. Incluso utilizando entornos de computación paralela, entrenar una red en ImageNet puede llevar semanas [17]. Adicionalmente, el consumo energético derivado de la IA es un tema de creciente preocupación dado el rápido aumento en la adopción y complejidad de estas tecnologías. Este consumo energético puede ser considerable, especialmente en el caso de los modelos de DL, que requieren una gran cantidad de recursos computacionales para su entrenamiento y

funcionamiento. En respuesta a esta problemática, ha emergido un nuevo enfoque dentro de la IA “Green AI” [18, 19] que se centra en desarrollar tecnologías y técnicas de IA que sean energéticamente eficientes y que minimicen su impacto ambiental. Reducir los tiempos de ejecución desempeña un papel crucial en el cumplimiento de los objetivos de la “Green AI”. Al disminuir la duración de los procesos computacionales, se reduce el consumo de energía necesario para realizar estas tareas.

Como se describió en la introducción general del TFG, el presente trabajo tiene como objetivo analizar una nueva metodología para el entrenamiento de CNN. Este estudio tiene como punto de partida el trabajo de Michael Mathieu, Yann LeCun y Mikael Henaff [10], que tomaremos como referencia. Este introduce un enfoque alternativo para realizar la operación de convolución [1], eje clave de las CNN. El método propuesto consiste en ejecutar la operación de convolución en el dominio de Fourier vía la FFT [6], lo que podría representar un avance significativo en términos de eficiencia computacional.

Existen trabajos anteriores a este, por ejemplo ya en los años 90 se exploró la posibilidad de utilizar la FFT para acelerar la inferencia en la primera capa de una red entrenada [11]. Sin embargo, esto no se utilizó durante el entrenamiento, posiblemente porque el número de mapas de características en ese momento era demasiado reducido como para que valiera la pena el esfuerzo de calcular las FFT en cada iteración. No obstante, dado el aumento en el número de mapas de características en las CNN modernas, resulta relevante estudiar esta línea de investigación.

En la primera parte de esta memoria, dedicada a la parte matemática del TFG, se desarrolló un marco teórico que proporcionó una introducción detallada y sistemática de algunos conceptos clave tales como la convolución, la Transformada de Fourier, y sus propiedades junto con resultados fundamentales como el Teorema de Convolución. Inspirados por estos, en esta segunda parte del trabajo, dedicada a la parte informática del TFG, se presentarán los resultados análogos correspondientes al contexto discreto, los cuales son esenciales para una comprensión integral del nuevo método propuesto. Además se realizará un análisis de eficiencia computacional del nuevo método propuesto con el propósito de compararlo con el enfoque tradicional, para lo que será imprescindible estudiar la eficiencia del algoritmo de la FFT. Posteriormente, se realizará un análisis experimental que evaluará tanto la nueva operación de convolución como su aplicación en problemas prácticos.

En esta parte, también se explorará un interesante tema de investigación propuesto como trabajo futuro en el artículo de referencia: la posibilidad de entrenar una CNN íntegramente en el dominio de la frecuencia. Este enfoque será examinado mediante su implementación en un contexto real, aplicándolo al problema de reconocimiento de dígitos manuscritos [27].

Para profundizar en el entendimiento del problema, a continuación se ofrece un análisis exhaustivo del mismo, junto con los objetivos trazados.

7.1. Descripción del Problema Planteado

La operación de convolución, que se describe en detalle matemáticamente en el Capítulo 5, es una técnica fundamental que evalúa cómo una función puede influir o modificar otra. En el campo de la VC, esta operación es clave para el filtrado de imágenes, permitiendo la detección y realce de características específicas dentro de una imagen. Típicamente la operación involucra una

7.1. DESCRIPCIÓN DEL PROBLEMA PLANTEADO

imagen de entrada y un determinado filtro o kernel, que es una matriz de dimensión reducida, diseñada para ser sensible a tipos particulares de características visuales. La operación de convolución, usando el **método tradicional**, implica deslizar el filtro o *kernel* sobre toda la imagen entrada y calcular la suma de productos en cada posición del filtro sobre la imagen, generando una matriz de salida, conocida como mapa de características.

Este mapa resalta las áreas donde las características específicas detectadas por el filtro son más prominentes, permitiendo así una identificación efectiva de patrones, texturas o bordes relevantes en la imagen original. Por ejemplo, algunos filtros pueden estar orientados a detectar bordes verticales, mientras que otros pueden enfocarse en bordes horizontales o en transiciones de color, y el mapa de características generado en cada caso será distinto según la respuesta de la imagen al filtro específico. Esta operación se describe en detalle en la Sección 11.4. En la Figura 7.1 se ilustra un ejemplo del proceso de convolución.

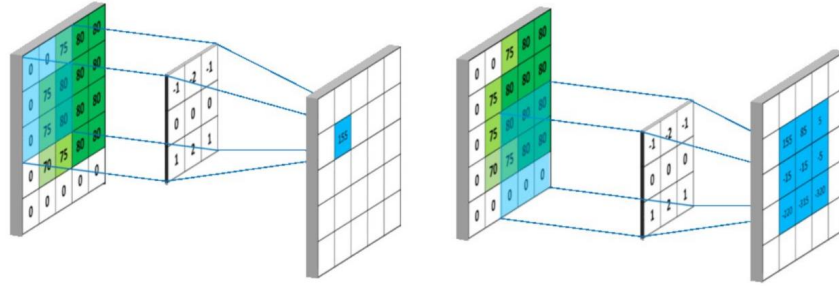


Figura 7.1.: Convolución de una imagen con un *kernel* de detección de bordes horizontales. Imagen modificada de [5].

La eficiencia del algoritmo de convolución es crucial, especialmente en aplicaciones de procesamiento de imágenes y aún más en aplicaciones de VC. La convolución directa, donde se desliza el filtro sobre cada posible posición en la imagen, tiene una complejidad computacional que puede ser alta, especialmente para imágenes y filtros de grandes dimensiones. La complejidad es típicamente de

$$O((n - k + 1)^2 \times k^2)$$

donde $n \times n$ es la dimensión de la imagen y $k \times k$ es la del filtro y la función $O(\cdot)$ es la cota superior asintótica comúnmente usada en análisis de complejidad algorítmica.

Por lo tanto, el primer problema que se abordará en esta segunda parte del TFG consistirá en explorar una alternativa al tradicional algoritmo de convolución descrito anteriormente. Para ello, se emplea el Teorema de Convolución, que se describe en detalle en el Capítulo 5. Este teorema es fundamental en el contexto discreto ya que permite interpretar el filtrado de imágenes en términos de frecuencias vía la Transformada de Fourier Discreta (DFT) que transforma la información desde el dominio espacial al dominio de la frecuencia. Todo esto se detalla en el Capítulo 11. El Teorema de Convolución afirma que la DFT de la convolución de una imagen I y un kernel K puede expresarse como el producto puntual de las DFTs de la imagen transformada $\hat{I}$ y del kernel transformado $\hat{K}$:

$$\widehat{I * K}(y) = \hat{I}(y) \cdot \hat{K}(y),$$

donde $*$ denota la operación de convolución. De este modo, al aplicar la Transformada de

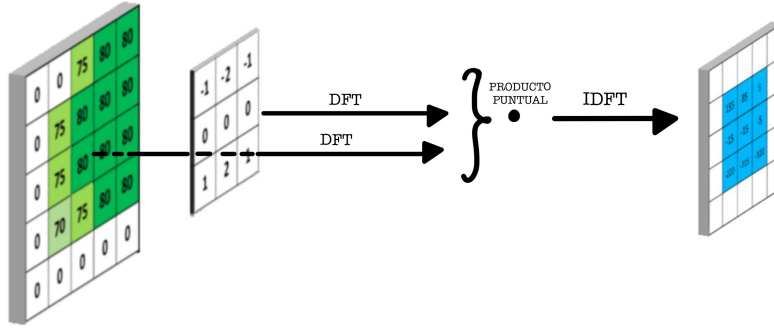


Figura 7.2.: Convolución de una imagen con un kernel de detección de bordes horizontales usando el Teorema de Convolución. Imagen modificada de [5].

Fourier Inversa (IDFT), se puede recuperar la convolución entre la imagen y el núcleo, siendo por tanto la convolución en el dominio espacial equivalente en cierto sentido al producto en el dominio de la frecuencia (dominio de Fourier). Véase la Figura 7.2.

Este entendimiento avanzado del procesamiento de imágenes en el dominio de la frecuencia, gracias al Teorema de Convolución, también subyace en tecnologías más sofisticadas como las CNN. Estas redes, detalladas en la Sección 9.4, incorporan la operación de convolución en su estructura, concretamente en la capa convolucional. La operación de convolución en estas redes es esencial para extraer patrones y características relevantes de los datos de entrada. De hecho, a nivel computacional, las operaciones significativas en el entrenamiento de CNN pueden considerarse como convoluciones entre pares de matrices bidimensionales, que pueden representar mapas de características de entrada y salida, gradientes de la pérdida con respecto a los mapas de características, o núcleos de pesos. Así, este nuevo método de realizar la convolución puede repercutir de manera directa en la eficiencia general del proceso de entrenamiento. Esta idea es la base del artículo [10], que propone una estructura para la capa convolucional ilustrada en la Figura 7.3.

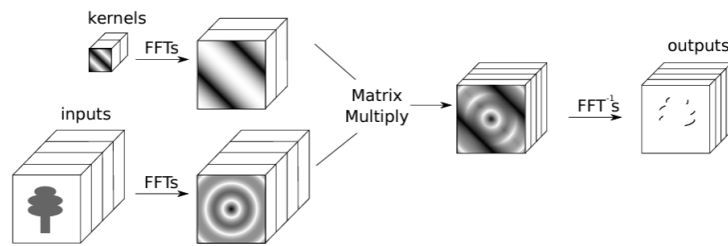


Figura 7.3.: Modelo propuesto basado en DFT. Imagen adaptada de [10].

En esta estructura, las convoluciones se realizan como productos en el dominio de Fourier. Cabe destacar que, aunque el producto puntual es más eficiente que la convolución directa, el algoritmo propuesto por el Teorema de Convolución requiere calcular la DFT de cada uno de los elementos involucrados en la operación de convolución y posteriormente aplicar la IDFT al resultado. No obstante, el artículo sugiere una optimización significativa: reutilizar los mapas

de características transformados varias veces. Así, al calcular las DFTs de las matrices solo una vez por conjunto, se facilita la realización de todas las convoluciones de manera eficiente mediante productos por pares.

Sin embargo, esta aproximación todavía requiere aplicar la IDFT al resultado en cada convolución. Por este motivo, en la sección de trabajos futuros del artículo, se plantea la posibilidad de realizar el entrenamiento completamente en el dominio de Fourier, de modo que no sea necesario recurrir a la IDFT en cada etapa. Este será el segundo problema que se abordará en este trabajo, el cual culminará con el análisis experimental de una CNN en el dominio de la frecuencia, diseñada para resolver el problema de reconocimiento de dígitos manuscritos.

7.2. Objetivos

El objetivo principal de esta segunda parte del TFG es desarrollar y evaluar un enfoque novedoso para el entrenamiento de CNN. Este objetivo consta de una primera etapa, donde se lleva a cabo un análisis experimental del método de convolución descrito en la sección anterior, y de una segunda, donde se diseña una CNN que realice el entrenamiento íntegramente en el dominio de Fourier usando ese método. A su vez, podemos identificar los siguientes objetivos parciales, cada uno de los cuales está asociado con un capítulo específico de esta segunda parte de la memoria.

- Estudiar el estado del arte en el campo del AA, específicamente en las CNN para obtener una comprensión profunda de su arquitectura. **Capítulo 9.**
- Analizar el dominio de la frecuencia a través de la DFT y describir la operación de convolución y sus propiedades. Presentación del Teorema de Convolución en el marco discreto: **Capítulo 11.**
- Analizar la eficiencia del algoritmo de la FFT, evaluar su importancia y aplicabilidad en la resolución del problema planteado: **Capítulo 12.**
- Analizar la eficiencia del método de convolución propuesto, en comparación con el método tradicional: **Capítulo 13.**
- Realizar un análisis del coste computacional de ejecutar el entrenamiento de una CNN en el dominio de la frecuencia y una comparativa con el coste computacional de los métodos de entrenamiento más habituales para CNN: **Capítulo 13.**
- Implementar el algoritmo de convolución utilizando el método propuesto y su aplicación en problemas prácticos: **Capítulo 14.**
- Implementar una CNN en el dominio de la frecuencia para abordar el problema de reconocimiento de dígitos manuscritos MNIST [27] y Fashion-MNIST [28]: **Capítulo 14.**

Capítulo 8.

Planificación

8.1. Metodología del Diseño

El aspecto más relevante del proyecto no es tanto el desarrollo del software, sino la presentación de una nueva línea de investigación en informática que se explora en primer plano a nivel teórico. El software actúa como una herramienta para ilustrar y aplicar los conceptos teóricos desarrollados, así como su aplicación a un problema informático real.

Desde el inicio, el desarrollo del proyecto fue planeado mediante un modelo en cascada, que característicamente impide retroceder entre etapas. No obstante, se preveía que debido a la novedad del problema a tratar, podrían surgir reajustes necesarios del mismo según los resultados de las pruebas experimentales. Por esta razón, se eligió un modelo de ciclo de vida en cascada con retroalimentación, cuyo esquema se puede observar en la Figura 8.1.

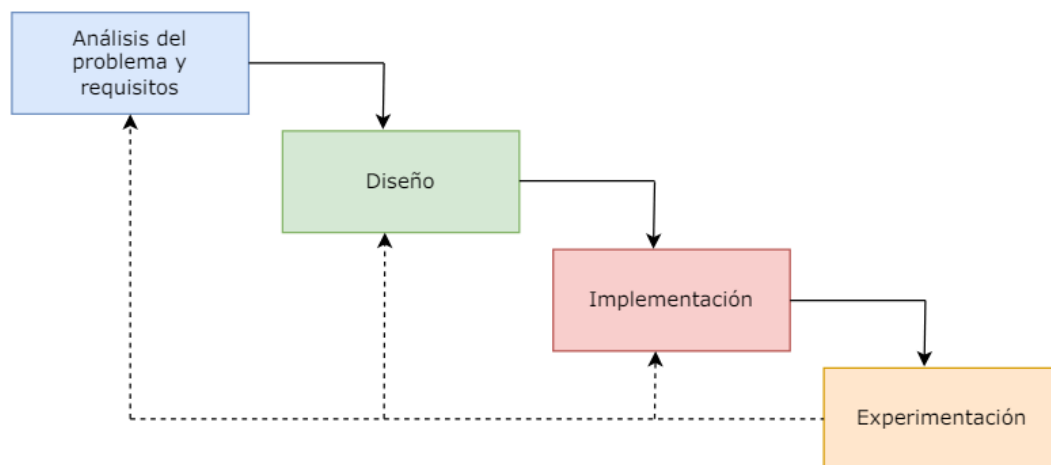


Figura 8.1.: Diseño en cascada retroalimentado empleado.

A continuación, se detallan las cuatro etapas que se seguirán:

1. **Análisis del problema y requisitos:** En esta primera etapa del proyecto, se profundizará en el contexto y la importancia del problema a resolver mediante un análisis exhaustivo

y una revisión del estado del arte de las soluciones existentes. Además, se llevarán a cabo reuniones con los tutores del trabajo para clarificar los requisitos y objetivos específicos del modelo a desarrollar.

2. **Diseño:** En esta fase del proyecto, se llevarán a cabo dos tareas principales:
 - a) Se implementará el algoritmo de la convolución en el dominio de Fourier.
 - b) Se abordará un problema real de AA en el dominio de Fourier.
3. **Implementación:** Todas las técnicas diseñadas en la etapa anterior serán implementadas.
4. **Experimentación:** Se pondrán en práctica todos los experimentos diseñados de forma teórica y se obtendrán resultados. En base a estos, se evaluará la posibilidad de retroceder en el ciclo de vida del proyecto para realizar las modificaciones necesarias.

8.2. Planificación Temporal

Dado el elevado volumen de trabajo durante el año académico, en el que un estudiante del doble grado cursa un total de 81 créditos, el desarrollo de este trabajo se inició en septiembre de 2023. Se comenzó con la parte matemática que está estrechamente vinculada con la informática, lo cual llevó a abordar también esta última desde una etapa temprana. A continuación, se detalla en la Figura 8.2 la planificación inicial correspondiente únicamente a esta segunda parte del presente trabajo.

Resalta a simple vista la gran cantidad de semanas invertidas en la primera etapa. Esto va en consonancia con lo comentado anteriormente acerca de que el proyecto tenía un fundamento teórico amplio que era necesario asentar para desarrollar el software. Por último, si se atiende a la planificación final del proyecto detallada en la Figura 8.3, cabe también destacar la vuelta atrás en el ciclo de vida a mediados de abril con el fin de revisar la etapa de análisis del problema y requisitos, y consultar a los tutores sobre los resultados obtenidos para progresar nuevamente hacia el resto de etapas.

8.3. Planificación Económica

Se estima que se han invertido alrededor de 15 horas semanales para abordar esta segunda parte de la memoria, lo que deja un total de 480 horas. Para la estimación del coste total del proyecto se asume un salario de 35 €/hora para un científico de datos de una empresa de base tecnológica, por lo que a partir de las horas invertidas, el trabajo tendría un presupuesto inicial de 16.800 €. Además, se deberían de incluir los costes de los recursos utilizados, como la conexión a internet, el consumo eléctrico y los dispositivos empleados, como ordenadores portátiles o la suscripción a Google Colab. El desglose del coste se encuentra en la Tabla 8.1

8.3. PLANIFICACIÓN ECONÓMICA

Recurso	Coste
Personal	16.800 €
Dispositivos	1.000 €
Suministro eléctrico	100 €
Acceso a internet	100 €
Total	18.000 €

Tabla 8.1.: Desglose del coste del proyecto.

Etapa	Noviembre				Diciembre				Enero				Febrero				Marzo				Abril				Mayo				Junio			
Análisis																																
Diseño																																
Implementación																																
Experimentación																																

Figura 8.2.: Descripción de la planificación inicial de las etapas del proyecto.

Etapa	Noviembre				Diciembre				Enero				Febrero				Marzo				Abril				Mayo				Junio			
Análisis																																
Diseño																																
Implementación																																
Experimentación																																

Figura 8.3.: Descripción de la planificación final de las etapas del proyecto.

Capítulo 9.

Estado del Arte

En este capítulo se revisan algunos conceptos fundamentales dentro del AA. En particular, se estudiará la arquitectura de las CNN, lo cual es esencial para plantear modificaciones a dicho modelo, siendo este uno de los objetivos principales de esta segunda parte informática del TFG. Para lograrlo, se ha aprovechado tanto los conocimientos adquiridos en cursos académicos, especialmente en VC y AA, como diversos artículos relevantes que se mencionarán donde sea conveniente.

9.1. Aprendizaje Automático

Actualmente, la IA constituye un campo crucial dentro de la informática, cuyo objetivo es proporcionar a los ordenadores habilidades para razonar o resolver problemas de manera inteligente. Dentro de este ámbito, la IA ha desarrollado una variedad de métodos que contribuyen a este fin, formando un extenso espectro de investigación. Algunos de los enfoques más destacados son, por ejemplo, el estudio de las Metaheurísticas, la Ingeniería del Conocimiento, y, más recientemente, el AA. Este trabajo se inscribe en esta última rama, y por lo tanto, se presentan a continuación diversas definiciones proporcionadas por diferentes expertos en la materia.

La primera definición reconocida de AA fue proporcionada por Arthur Samuel en 1959 [8]. Samuel definió el AA como “*el campo de estudio que confiere a los ordenadores la capacidad de aprender sin estar explícitamente programados*”. Esta definición es notable por capturar el objetivo principal: que una máquina “aprenda” a solucionar un problema específico a partir de la experiencia, en vez de limitarse a seguir instrucciones preestablecidas. Una definición más reciente proporcionada por Tom Mitchell en 1998 [9] establece que “*un programa informático aprende de la experiencia E en relación con una tarea T y un criterio de rendimiento P , si su rendimiento en T , medido por P , mejora con la experiencia E .*” Esta definición es más concreta que la anterior ya que nos permite identificar los componentes esenciales para abordar un problema mediante técnicas de AA. Estos componentes incluyen un problema específico (T) que se desea resolver utilizando un ordenador, una experiencia (E) relacionada con esa tarea, que comúnmente se representa como una base de datos o conjunto de observaciones y un criterio de rendimiento (P) que evalúa la efectividad del algoritmo en resolver el problema y suele estar vinculado a una función objetivo que se busca optimizar o minimizar.

Considerando el conjunto de datos disponibles para el entrenamiento de los modelos y el problema que se va a resolver se presenta la siguiente clasificación, según [29] en:

- **Aprendizaje Supervisado.** Los algoritmos de AA utilizados en este conjunto se ca-

racterizan por disponer de una base de datos etiquetada. Esto significa que para cada entrada x , conocemos su etiqueta correspondiente y . El objetivo principal es determinar la función f que establece la relación entre x y y , de modo que $f(x) = y$.

- **Aprendizaje No Supervisado.** Los algoritmos de Aprendizaje No Supervisado se distinguen por trabajar con datos que no están etiquetados, es decir, el modelo no cuenta con salidas específicas predefinidas. En lugar de predecir resultados, el objetivo es explorar y descubrir características inherentes al conjunto de datos. Estos modelos se enfocan en identificar patrones, formar agrupaciones y detectar anomalías dentro de los datos de entrada.
- **Aprendizaje por Refuerzo.** El aprendizaje por refuerzo es una técnica de AA en la que el modelo aprende a través de la toma de decisiones mediante la experimentación con un entorno interactivo. El modelo recibe recompensas o penalizaciones basadas en las acciones que realiza, guiándolo para formular una estrategia que maximice la suma de las recompensas a lo largo del tiempo.

En la sección informática de este TFG, la solución propuesta al problema planteado se sitúa dentro de los modelos del **Aprendizaje Supervisado**. Por esta razón, continuaremos profundizando en este enfoque y en los siguientes cuatro aspectos fundamentales para construir un algoritmo en AA:

- **Conjunto de Datos.** Este debe ser suficientemente grande para facilitar el entrenamiento efectivo del modelo, incluyendo etiquetas en casos de aprendizaje supervisado. Se divide en un conjunto de entrenamiento, usado para el aprendizaje del modelo, y un conjunto de prueba, destinado a evaluar el rendimiento del modelo con datos nuevos.
- **Función de Pérdida.** Esta función, también conocida como función objetivo o de coste, mide la discrepancia entre las predicciones del modelo y los valores reales. La selección de la función de pérdida se basa en el tipo de problema y las características de los datos.
- **Procedimiento de Optimización.** Es el proceso encargado de minimizar la función de pérdida mediante el ajuste de los parámetros del modelo.
- **Modelo.** Representa la arquitectura del algoritmo, compuesta por capas interconectadas que procesan los datos de entrada para generar estimaciones o predicciones.

9.1.1. Aprendizaje Supervisado

En el campo del Aprendizaje Supervisado, destacan dos tareas esenciales: Clasificación y Regresión, que se diferencian significativamente por la naturaleza de los valores que se intentan predecir. A continuación se presentan ambas:

9.1.1.1. Regresión

La tarea de regresión [30] busca predecir valores continuos. En este tipo de problemas, el objetivo es determinar la función f que asocie cada dato con su correspondiente etiqueta.

$$f(x) = y \text{ con } x \in \mathbb{R}^n, y \in \mathbb{R}^m.$$

Generalmente, obtener la función f es una tarea de una alta complejidad, por lo que se busca aproximar dicha función mediante otra g , perteneciente generalmente a una familia de funciones parametrizadas que se elige y se entrena a partir de los datos etiquetados proporcionados.

La función de coste típicamente utilizada para este tipo de problemas es el *error cuadrático medio* [31], que evalúa la precisión con la que la función g aproxima a f .

$$\text{ECM} = \frac{1}{n} \sum_{i=1}^n (f(x_i) - g(x_i))^2.$$

9.1.1.2. Clasificación

En las tareas de clasificación [30], el objetivo es predecir valores discretos. Esto significa que el modelo debe aprender a reconocer patrones en los datos de entrenamiento que le permitan asignar una categoría o clase específica de entre un conjunto predefinido de clases $\{k_1, \dots, k_m\}$, a cada entrada según sus características.

$$f(x) = k \text{ con } x \in \mathbb{R}^n, k \in \{k_1, \dots, k_m\}.$$

Este proceso implica analizar las características de una instancia desconocida y emplear el modelo aprendido para determinar a qué clase pertenece. Los casos más sencillos de este problema son los de *clasificación binaria*, y en ellos se pretende agrupar los datos en dos posibles clases que suelen codificarse como 0 y 1. La función de coste típicamente utilizada para este tipo de problemas es la *entropía cruzada* [31] que mide la diferencia entre las distribuciones de probabilidad reales y las predicciones del modelo.

El problema real que se abordará en esta memoria siguiendo el enfoque planteado anteriormente, corresponde con un problema de **clasificación**.

9.1.2. Gradiente Descendente

Hasta el momento se ha expuesto qué es el AA y cómo se formalizan sus problemas para poder resolverlos, presentando diferentes tipos de tareas junto con algunas funciones de pérdida asociadas a cada una. Sin embargo, todavía no se ha explorado ningún algoritmo específico utilizado en la minimización de la función de pérdida. Por esta razón, a continuación, describiremos uno de los algoritmos empleados: el Gradiente Descendente [32].

El Gradiente Descendente es un algoritmo fundamental en optimización que se basa en la idea de que el gradiente de una función indica la dirección del aumento más rápido de la misma. Por lo tanto, al moverse en la dirección opuesta al gradiente, es posible aproximarse al mínimo de la función. Dada una función de pérdida L , el algoritmo podría describirse como en Algoritmo 1.

Debido a que la búsqueda lineal es computacionalmente costosa, pues implica evaluar la función repetidas veces, se opta por utilizar un parámetro denominado tasa de aprendizaje (learning rate, lr) μ . Este parámetro determina el “tamaño” del paso que se toma en cada iteración hacia el mínimo, simplificando el proceso y reduciendo la carga computacional.

$$\theta_{t+1} = \theta_t - \mu \Delta(\theta).$$

Algoritmo 1 Gradiente Descendente

Dado un punto inicial $\theta \in \text{Dom}(L)$

Repetir

$\Delta\theta := -\nabla L(\theta)$

Búsqueda lineal. $t^* = \arg \min_{t>0} L(\theta + t \cdot \Delta\theta)$

Actualizar. $\theta := \theta + t^* \Delta\theta$

Hasta que el criterio de parada se cumpla.

Es crucial seleccionar adecuadamente la tasa de aprendizaje, ya que un valor demasiado bajo puede resultar en una convergencia lenta hacia el mínimo, requiriendo muchas iteraciones. Por otro lado, un learning rate excesivamente alto puede impedir la convergencia al provocar oscilaciones. Véase la Figura 9.1, donde el concepto de época se refiere a una iteración completa sobre el conjunto de datos de entrenamiento completo. Si el método del gradiente descendente converge, lo hace al mínimo global de la función de coste cuando esta presenta una función de pérdida convexa. No obstante, si la función carece de convexidad, el método podría terminar alcanzando un mínimo local en vez del global.

El Gradiente Descendente utilizando mini lotes, comúnmente conocido como Gradiente Descendente Estocástico, [33, 34, 35] es una variante del tradicional algoritmo del Gradiente Descendente que implementa una estrategia de optimización estocástica. En lugar de utilizar el gradiente completo, calculado a partir de todos los datos disponibles, el SGD emplea una estimación del gradiente obtenida a partir de un subconjunto aleatorio de datos. Este enfoque es particularmente útil en problemas de optimización de alta dimensión, donde puede reducir significativamente la carga computacional y acelerar las iteraciones, aunque a costa de una tasa de convergencia más lenta.

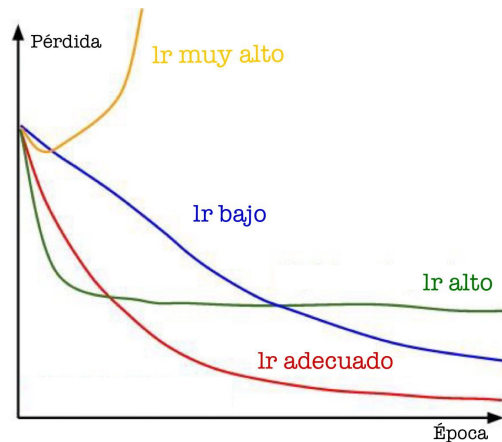


Figura 9.1.: Gráfica de la tasa de aprendizaje lr. Imagen modificada de [36]

9.2. Visión por Computador

La VC es un campo interdisciplinario que combina entre otras áreas la IA y AA con el objetivo de procesar imágenes mediante ordenadores, permitiendo que estas extraigan información de

manera similar a como lo haría un ser humano [2].

A continuación, se exploran otras definiciones y perspectivas sobre este concepto ofrecidas por distintos autores, las cuales enriquecen y amplían nuestra comprensión del tema desde varios enfoques:

David A. Forsyth y Jean Ponce muestran la VC como un campo interdisciplinar: *“En visión por computador, se utilizan métodos estadísticos para desentrañar datos mediante modelos que se construyen con la ayuda de la geometría, la física y la teoría del aprendizaje.”* [3]. En [4] Russel y Norvig describen la VC como el proceso de *“recuperar información del aluvión de datos que provienen de los ojos o cámaras”*. Por último, el científico Szeliski añade en [37] que *“En VC intentamos describir el mundo que vemos en una o más imágenes y reconstruir sus propiedades, como la forma, la iluminación y las distribuciones de color”*.

Dentro de los problemas clásicos de este campo se encuentran el reconocimiento de objetos o personas en imágenes, la segmentación y la clasificación. Recientemente, esta área ha recibido un impulso significativo en la comunidad científica gracias al desarrollo del DL las CNN profundas, las cuales serán explicadas con más detalle en la próxima sección. Estas innovaciones han facilitado la creación de programas altamente eficaces en el procesamiento de imágenes.

9.3. Aprendizaje Profundo

El DL [7] es una disciplina dentro del campo de la IA que se especializa en el desarrollo y entrenamiento de modelos de Redes Neuronales artificiales caracterizados por tener múltiples capas y una alta complejidad estructural.

En contraste con los modelos convencionales de AA que aprenden a partir de los datos de manera general, las redes neuronales en el DL son significativamente más complejas, donde cada capa de la red se encarga de extraer y procesar características cada vez más abstractas y sofisticadas de la información proporcionada. La figura 9.2 muestra de manera muy ilustrativa estos dos enfoques.

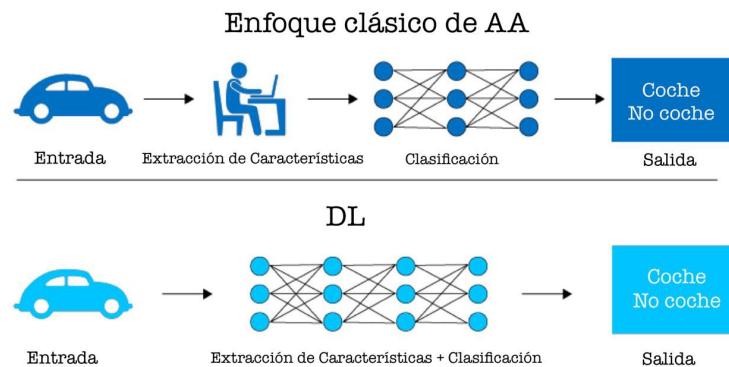


Figura 9.2.: Comparación entre DL y el enfoque clásico de AA. Imagen modificada de [38].

De manera más general, la Figura 9.3 presenta un diagrama exhaustivo que ilustra la integración y relación entre los campos mencionados y otras disciplinas adyacentes. Este diagrama es fundamental para entender cómo diversas áreas del conocimiento científico y tecnológico

convergen y se complementan. En el diagrama se puede ver reflejado como este trabajo se encuentra en la intersección de las áreas de VC y DL. Concretamente, en el ámbito de las CNN, que se detallarán más adelante tras introducir otros conceptos.

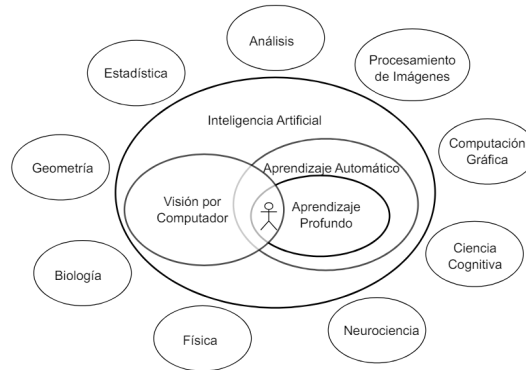


Figura 9.3.: Diagramas de inclusión de los campos de IA, AA, DL y otras áreas relacionadas. Imagen modificada de la asignatura de VC.

9.3.1. Feedforward Neural Network

La arquitectura más elemental en DL se denomina Feedforward Neural Network (FNN) o perceptrón multicapas (MLP) [39] y constituye la base para un número elevado de soluciones que emplean DL en su implementación.

Las FNN al igual que como se describió anteriormente, tienen por objetivo aproximar una determinada función f que asocia a cada entrada x una salida y . Para llevar a cabo esta tarea se construye otra función g de manera que $g(x) = \tilde{y}$ sea el valor predicho para esa entrada x , donde w corresponde con unos parámetros que se pretende aprender para aproximar mejor la función f . Por ejemplo, en el esquema que aparece en la Figura 9.7 se tiene que g sigue un modelo lineal que se define como

$$g(x_1, x_2) = w_0 + w_1x_1 + w_2x_2$$

El entrenamiento del modelo [40] por tanto, consistirá en determinar los pesos w_i que resultan en la mejor aproximación de la función f , usando para ello algoritmos basados en Gradiente descendente. De esta manera, se convierte el problema en uno de optimización de una determinada función de pérdida. A continuación se detallan algunos aspectos importantes sobre las FNN.

- Los algoritmos empleados por las redes se denominan **feedforward** porque la información fluye desde la entrada x a través de una serie de operaciones que definen g hasta llegar a la salida $\tilde{y}$.
- Las FNN se denominan **redes** porque generalmente se representan componiendo juntas diversas funciones diferentes asociadas cada a una a una capa. Además, gráficamente el modelo está asociado con un grafo acíclico dirigido que describe cómo se componen las funciones juntas y que forma una especie de “red”. En el ejemplo de la Figura 9.4, la

FNN podría describirse como $g = g^4(g^3(g^2(g^1(x))))$ donde g^1 es la primera capa, g^2 la segunda, g^3 la tercera y finalmente g^4 es la capa de salida.

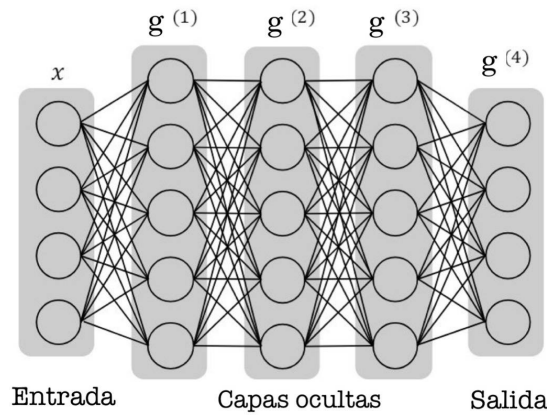


Figura 9.4.: Ejemplo de una FNN. Imagen modificada de [41].

- Los ejemplos de entrenamiento especifican directamente lo que debe hacer la capa de salida en cada punto, esta debe producir un valor “cercano” a su etiqueta. El comportamiento de las otras capas intermedias no está directamente especificado por los datos de entrenamiento, por lo que el algoritmo de aprendizaje debe decidir cómo utilizar esas capas para producir la salida deseada. Por ello, se les denomina **capas ocultas**.
- La **profundidad** de una red en DL se define por el número de capas ocultas que posee, lo que explica el término profundo en la denominación del término, debido a la presencia de múltiples capas ocultas.
- Las FNN tienen el término **neuronal** en su denominación porque están inspiradas de manera laxa en la neurociencia. Cada capa oculta de la red es típicamente vectorial. La dimensionalidad de estas capas ocultas determina el **ancho** del modelo. Cada elemento del vector puede interpretarse como jugando un papel análogo al de una neurona. En lugar de pensar en la capa como representando una única función de vector a vector, también se puede pensar en una capa como aquella consistente en muchas unidades que actúan en paralelo, cada una representando una función de vector a escalar 9.5. Cada unidad se asemeja a una neurona en el sentido de que recibe entrada de muchas otras unidades y calcula su propio valor de activación. La idea de usar muchas capas de representaciones vectoriales se deriva de la neurociencia.

Para que la red sea capaz de aproximar funciones objetivo no lineales, estas combinaciones lineales como la de g no son suficientes. Por ello, se incorporan funciones de **activación no lineales**. Estas funciones permiten a la red captar y modelar la complejidad de los datos de entrada.

Entre las funciones de activación más utilizadas se encuentran las siguientes [42]:

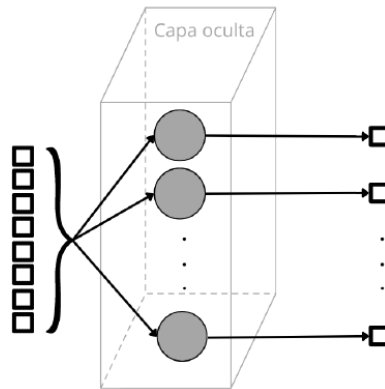


Figura 9.5.: Representación de una capa oculta en la cual se visualiza un conjunto de funciones que tienen un dominio vectorial y una imagen escalar.

- **Función Sigmoide.** Se utiliza típicamente para la clasificación binaria, se define como

$$\sigma(x) = \frac{1}{1 + e^{-x}}.$$

Transforma valores reales en el rango $(0, 1)$. El principal problema de esta función es que si no se manejan adecuadamente la inicialización de los pesos o el preprocesamiento de los datos, estas no linealidades pueden 'saturarse' y detener completamente el aprendizaje, esto da lugar al desvanecimiento del gradiente. Este fenómeno ocurre cuando los valores de entrada se alejan demasiado de cero, llevando a la función de activación a aproximarse a sus límites asintóticos, donde su pendiente tiende a ser casi cero. Cuando la función se satura, las derivadas parciales, esenciales para el cálculo del gradiente, se vuelven también muy pequeñas, cercanas a cero.

- **Función Tangente Hiperbólica.** Es similar a la función sigmoide, se define matemáticamente como

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}.$$

Transforma valores reales en el rango $[-1, 1]$, proporcionando una salida centrada alrededor de cero. El desvanecimiento del gradiente es un problema que también está presente en esta función de activación.

- **Función ReLU.** Es una de las funciones de activación más utilizadas (junto con algunas de sus variaciones) en redes profundas ya que de forma experimental se ha comprobado que permite un entrenamiento más rápido. Se define matemáticamente como

$$ReLU(x) = \max\{0, x\}.$$

Esta función en ciertos casos puede llevar al llamado problema de la "ReLU muerta". Esto sucede si una neurona ReLU se inicializa de tal manera que nunca se activa (porque recibe entradas negativas) o si los pesos de una neurona experimentan un cambio significativo

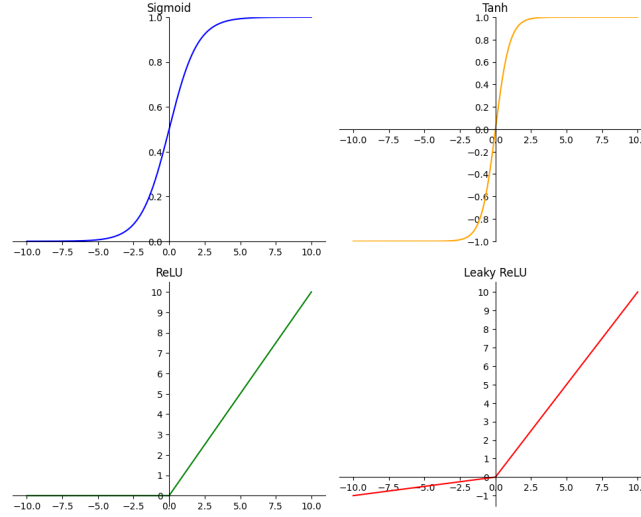


Figura 9.6.: Visualización de funciones de activación.

debido a una actualización considerable durante el entrenamiento que provoca que la suma ponderada de las entradas siga siendo negativa. Una vez que una neurona cae en este estado, se vuelve inútil para el resto del proceso de entrenamiento, similar a cómo un daño cerebral permanente puede dejar inactiva una parte del cerebro. Algunas variantes de ReLU, como Leaky ReLU o Parametric ReLU, ofrecen alternativas que permiten flujos de gradientes incluso para entradas negativas, reduciendo así el riesgo de neuronas muertas.

- **Función Leaky ReLU.** Para abordar el problema mencionado anteriormente con la ReLU, se ha desarrollado una variante conocida como Leaky ReLU. Esta introduce una pequeña pendiente positiva m para los valores negativos, permitiendo así un flujo continuo de gradientes durante el entrenamiento, incluso para las neuronas que de otro modo estarían inactivas. Se define matemáticamente como

$$ReLU(x) = \max\{mx, x\}.$$

- **Función SoftMax.** Se emplea para representar la distribución de probabilidad sobre n clases distintas, es decir, permite al modelo realizar una clasificación multiclase. Se define como

$$\text{Softmax}(x)_k = \frac{e^{z_k}}{\sum_{j=1}^n e^{z_j}} \text{ con } k = 1, \dots, n.$$

Dado un vector de entrada n -dimensional de valores reales, produce un vector n -dimensional de elementos en $(0, 1)$, por lo que la componente más próxima a 1 representaría a que clase pertenece el elemento en un problema de clasificación multiclase.

En la Figura 9.3.1 se muestran estas funciones de activación.

De manera que la función g^i que referencia la capa oculta i de la red, suele tener la siguiente

expresión

$$g^i(x) = \alpha(w^T x)$$

Donde x representa la entrada, w_i el vector de pesos correspondientes a la capa i de la red y α una función de activación. Al utilizar una función de activación no lineal, se habilita que la función adquiera una naturaleza no lineal, incrementando así su capacidad expresiva. Por lo que el esquema final de una capa oculta se vería reflejado en la Figura 9.7.

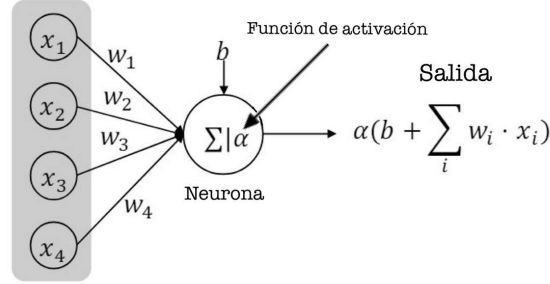


Figura 9.7.: Esquema de una capa oculta. Imagen modificada de [41]

9.3.2. Backpropagation

Como ya se mencionó, en las FNN la información fluye de manera unidireccional, sin procesos de retroalimentación. La entrada x proporciona la información inicial que se transmite a través de las capas ocultas hasta producir una salida. Durante el entrenamiento es crucial establecer un método que permita al modelo utilizar la información obtenida a través del gradiente. Todas las funciones de las capas intermedias de la red son derivables, por lo que se podría calcular de forma explícita su derivada en cada caso, lo que ocurre es que este proceso es costoso computacionalmente. En lugar de esto se aplica la técnica de Backpropagation [43], esta no debe confundirse con el Gradiente Descendente. Específicamente, el Backpropagation se refiere únicamente al método para calcular el gradiente, mientras que el algoritmo de Gradiente Descendente facilita el aprendizaje utilizando dicho gradiente. El algoritmo consta de dos fases fundamentales:

- Cálculo hacia adelante (Forward calculation): con el input inicial y una serie de parámetros definidos, la red neuronal calcula el valor de cada neurona de forma secuencial.
- Propagación hacia atrás (Backward propagation): la red determina el error en cada variable y actualiza los parámetros utilizando las derivadas parciales correspondientes, procediendo en orden inverso. La idea del algoritmo de Backpropagation es ir calculando la derivada en cada nodo del grafo computacional mediante la aplicación de la regla de la cadena. Por ejemplo, consideremos la función $f(x, y, z) = g(x, y)h(z)$. Si quisiéramos calcular la derivada parcial de f con respecto a x , aplicando la regla de la cadena, tendríamos que:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial g} \cdot \frac{\partial g}{\partial x}$$

Se presenta a continuación un ejemplo concreto de un caso sencillo.

Ejemplo 9.3.1. Se quiere computar el gradiente de la función $f(x, y, z) = (x + y)z$.

En particular, nótese que esta expresión se puede descomponer en dos :

$$q = x + y,$$

$$f = qz.$$

De donde se sigue que

$$\begin{aligned} \frac{\partial f}{\partial q} &= z, & \frac{\partial f}{\partial z} &= q, \\ \frac{\partial q}{\partial x} &= 1, & \frac{\partial q}{\partial y} &= 1. \end{aligned}$$

Sin embargo, el gradiente sobre el valor intermedio q no es de interés; el valor de $\frac{\partial f}{\partial q}$ no es útil por sí solo. En cambio, sí lo es el gradiente de f respecto a las entradas x, y, z . De la regla de la cadena se sigue que

$$\begin{aligned} \frac{\partial f}{\partial x} &= \frac{\partial f}{\partial q} \frac{\partial q}{\partial x}, \\ \frac{\partial f}{\partial y} &= \frac{\partial f}{\partial q} \frac{\partial q}{\partial y}. \end{aligned}$$

La Figura 9.8 presenta el resultado del ejemplo con valores específicos de entrada.

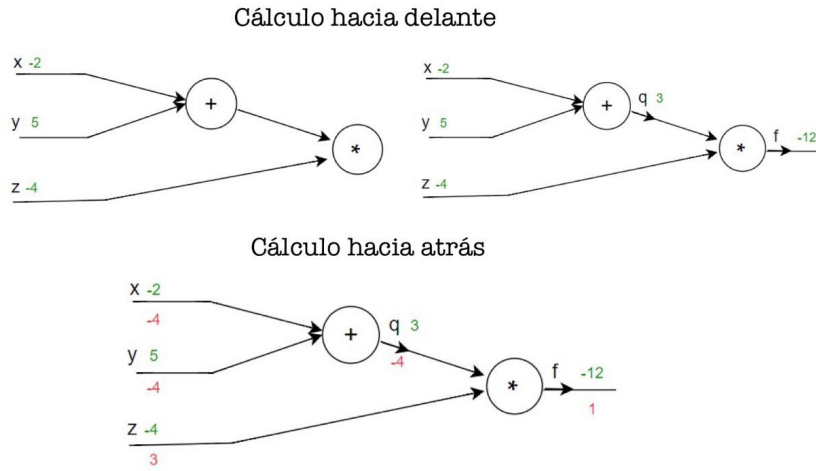


Figura 9.8.: Ejemplo del algoritmo de Backpropagation para $f = (x + y)z$.

Aunque esta expresión es lo suficientemente simple como para diferenciarla directamente, en casos más complejos, el uso del algoritmo de backpropagation se vuelve crucial para computar eficientemente las derivadas.

9.4. Redes Neuronales Convolucionales

Las CNN [7] constituyen una herramienta fundamental en DL diseñadas especialmente para procesar información que se presenta en forma de estructuras reticulares, como imágenes. Como su nombre indica estas redes utilizan el operador matemático conocido como **convolución** en al menos una de sus capas. Este operador, eje central de las CNN, se analiza exhaustivamente desde una perspectiva teórica en la primera sección de la memoria, específicamente en el Capítulo 5 donde también se incluyen ejemplos prácticos 5.5.3 que destacan su relevancia en el campo de VC, concretamente en el tratamiento de imágenes. El estudio en mayor profundidad de este operador en el ámbito de AA se puede encontrar en el Capítulo 11.

Cabe destacar los principales motivos por los que las CNN pueden dar lugar a modelos con mejores prestaciones que las FNN [39].

- **Interacciones dispersas o Conectividad local.** Dado que trabajar con entradas de grandes dimensiones, como las imágenes, hace impráctico conectar cada neurona con todas las de la capa anterior (como es típico en las redes neuronales clásicas) a través de la multiplicación matricial entre una matriz de parámetros y un parámetro independiente. En las CNN se opta por conectar cada neurona únicamente con una región local del volumen de entrada, esto se denomina conectividad local o interacción dispersa. Esto se logra mediante la reducción del tamaño del núcleo (conservando la misma profundidad), lo que permite centrarse en áreas más pequeñas y localizadas de la entrada, como se muestra en la Figura 9.9. Si hay m entradas y n salidas, entonces la multiplicación matricial requiere $m \times n$ parámetros, y los algoritmos utilizados en la práctica tienen un tiempo de ejecución de $O(m \times n)$. Si limitamos el número de conexiones que cada salida puede tener a k , entonces el enfoque de conectividad dispersa solo requiere $k \times n$ parámetros y un tiempo de ejecución de $O(k \times n)$. Por lo que se reduce el número de operaciones que se deben realizar durante los procesos de entrenamiento y evaluación del modelo al mismo tiempo que el tamaño de los datos que se deben almacenar.
- **Reparto de parámetros.** El compartir parámetros se refiere al uso del mismo parámetro para más de una función en un modelo. En una red neuronal tradicional, cada elemento de la matriz de pesos se utiliza exactamente una vez al calcular la salida de una capa; multiplicándose por un elemento de la entrada y luego no se reutiliza más. En una CNN cada elemento del kernel se utiliza en cada posición de la entrada (excepto quizás algunos píxeles de borde dependiendo de las decisiones de diseño que se tomen al respecto), de forma que se aprende un único conjunto de parámetros, tal y como se muestra en la Figura 9.10.
- **Equivariante con respecto de la traslación.** Una función es equivariante respecto de un operador, si al transformar la entrada mediante dicho operador, la salida cambia de la misma manera. Específicamente, una función $f(x)$ es equivariante a una función g si $f(g(x)) = g(f(x))$. En el caso de la convolución, si consideramos que g es cualquier función que traslada la entrada, es decir, la desplaza, entonces la función de convolución es equivariante a g . Esta propiedad se demostró en la primera parte de la memoria en la Proposición 5.3.3. Esto tiene una importante consecuencia ya que al crear la convolución un mapa 2D que muestra dónde aparecen ciertas características de un objeto de una imagen de entrada, si movemos el objeto en la entrada, su representación se moverá del mismo modo en la salida. Véase la Figura 9.11.

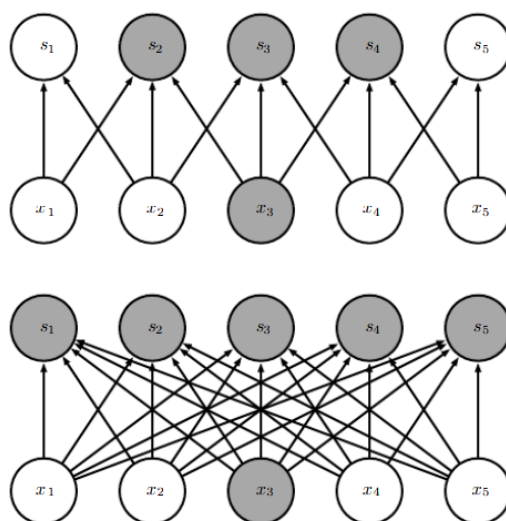


Figura 9.9.: En la imagen superior, la entrada x_3 interactúa con un kernel reducido a tres elementos, lo que afecta exclusivamente a las salidas s_2 , s_3 y s_4 . En cambio, en la imagen inferior, el kernel abarca el mismo tamaño que la entrada completa, resultando en que todas las salidas sean influenciadas por x_3 . Imagen obtenida de [39].

9.4.1. Arquitectura de una CNN

Se describe a continuación la estructura de una CNN, que se compone generalmente de tres tipos de capas: **capas convolucionales**, **capas de Pooling** y **capas totalmente conectadas**. La Figura 9.12 ilustra la estructura característica de una red neuronal convolucional (CNN).

9.4.1.1. Capa Convolucional

En una capa convolucional, varios filtros (también llamados núcleos o kernels) se aplican sobre la entrada deslizándose a través de ella para producir mapas de características mediante la operación de convolución discreta, que será estudiada en profundidad en el Capítulo 11. Esta operación consiste en calcular la suma ponderada de los valores en cada ubicación de la entrada que el filtro cubre, resultando en un mapa de características convolucionales. Véase la Figura 9.13. Es importante que la profundidad de los filtros coincida con la profundidad del volumen de entrada para asegurar que todas las dimensiones sean compatibles durante la convolución. La salida de la capa convolucional depende de tres parámetros: **depth**, **stride** y **padding**.

- **Parámetro depth.** La profundidad se refiere al número de filtros que se aplican a la entrada, el objetivo es que cada filtro adquiera conocimiento diferente sobre la información de entrada generando cada uno un mapa de activación. En una imagen en color RGB, la profundidad podría ser 3, correspondiendo a los canales Rojo, Verde y Azul.
- **Parámetro stride.** El stride indica el paso con el que el filtro se desplaza a lo largo de la entrada durante la operación de convolución. Un stride de 1 significa que el filtro se mueve de uno en uno por los píxeles mientras que un stride de 2 implica que

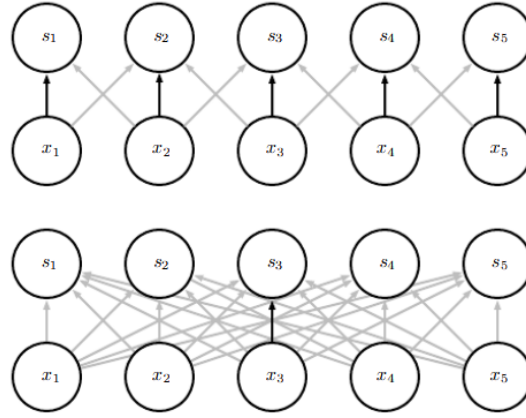


Figura 9.10.: En el modelo superior se utiliza el reparto de parámetros, las flechas negras indican el uso del elemento central de un núcleo de tres dimensiones, por tanto el mismo conjunto de parámetros se usa para todas las entradas y salidas. En el modelo inferior la única flecha negra indica el uso del elemento central de la matriz de pesos en una modelo totalmente conectado. Este modelo no tiene reparto de parámetros, el mismo conjunto de parámetros sólo puede utilizarse en la interacción entre la entrada x_3 y la salida s_3 . Imagen obtenida de [39].

el filtro se desplaza de dos en dos píxeles. Un stride mayor reduce el tamaño del mapa de características resultante, lo que puede ser útil para reducir la dimensionalidad de la salida y aumentar la eficiencia computacional de la red.

- **Parámetro padding.** El padding se refiere a la práctica de agregar píxeles adicionales alrededor de los bordes de la entrada antes de aplicar la operación de convolución. Este parámetro se usa para controlar la dimensión de salida de los mapas de activación, esto se hace para evitar que el tamaño de la entrada disminuya después de cada capa de convolución.

Finalmente teniendo en cuenta estos parámetros, es posible determinar el volumen de salida de una capa convolucional para un volumen de entrada de dimensión $W \times W \times d$ y empleando filtros de dimensión $F \times F \times d$, con padding P y stride S , la salida de la capa convolucional viene determinada por:

$$\frac{W - F + 2P}{S + 1}.$$

Por lo general, la salida de una capa de convolución se convierte en la entrada de una función de activación, tal como se describió en la sección anterior. La función ReLU es ampliamente utilizada para este propósito, aunque también es común emplear Leaky ReLU.

9.4.1.2. Capa de Pooling

Su función principal es reducir progresivamente la dimensión espacial (el ancho y el alto, no la profundidad) de la representación de entrada para disminuir la cantidad de parámetros y cálculos en la red. Generalmente se emplean filtros de dimensión 2×2 con un stride de 2 que reducen a la mitad la dimensión de la entrada manteniendo la profundidad.

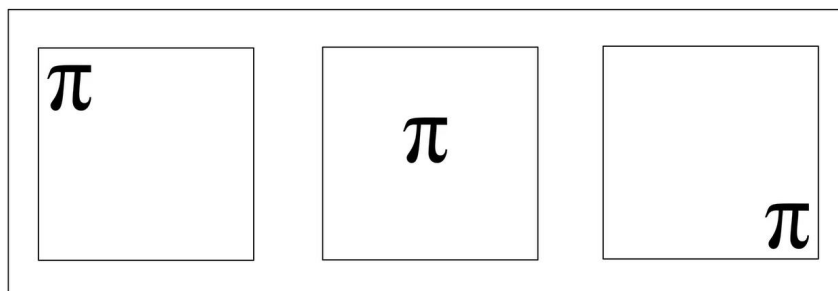


Figura 9.11.: Los tres números π deben identificarse como iguales, aunque se encuentren desplazados.

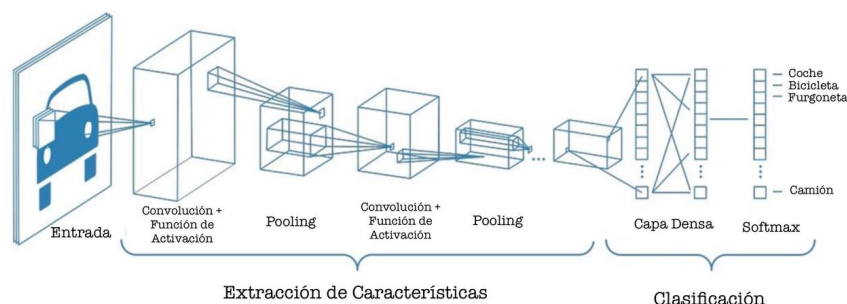


Figura 9.12.: Estructura típica de una CNN para un problema de clasificación. Imagen modificada de [44].

Así, deslizando una ventana de 2×2 sobre la salida de la capa anterior, esta técnica resume la información contenida en dicha salida aplicando un criterio específico a los elementos dentro del campo receptivo. Por ejemplo, se puede optar por seleccionar el máximo de estos elementos (**Max Pooling**) o calcular su promedio (**Average Pooling**). El pooling además de reducir la cantidad de datos a procesar, ayuda a la red a ser insensible a pequeñas traslaciones y distorsiones en la imagen de entrada. Véase la Figura 9.14.

9.4.1.3. Capa Totalmente Conectadas

Generalmente al final de una CNN suele encontrarse una capa totalmente conectada (FC), también conocida como capa densa y que es responsable de consolidar y transformar las características extraídas en predicciones finales. En esta capa cada neurona está conectada a todas las neuronas de la capa anterior, razón por la cual se le denomina totalmente conectada. La entrada de una capa totalmente conectada proviene de la salida de la última capa de Pooling o Convolutiva, esta se aplanar, transformando la salida en un vector que se conecta entonces a varias capas totalmente conectadas, las cuales tienen la misma estructura que una red neuronal clásica con una o dos capas ocultas generalmente y por lo tanto está diseñada específicamente para procesar vectores de entrada de una dimensión determinada. Debido a esta estructura, las capas totalmente conectadas suelen establecer el volumen de entrada requerido para el correcto

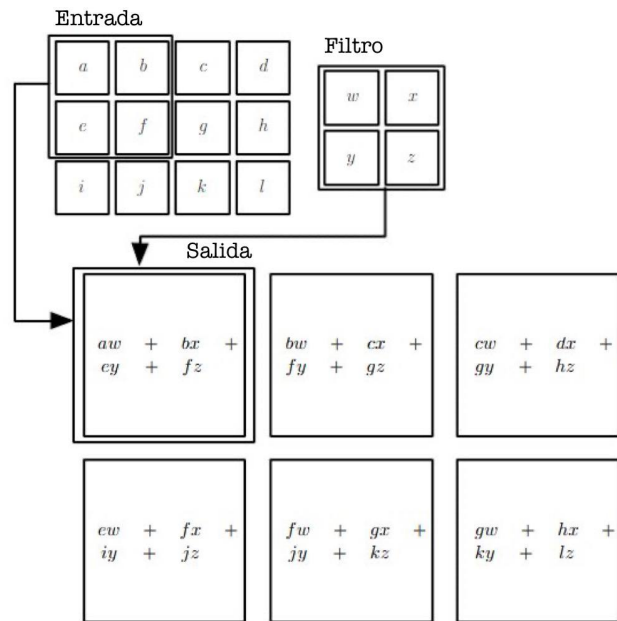


Figura 9.13.: Operación de convolución. Imagen modificada de [39].

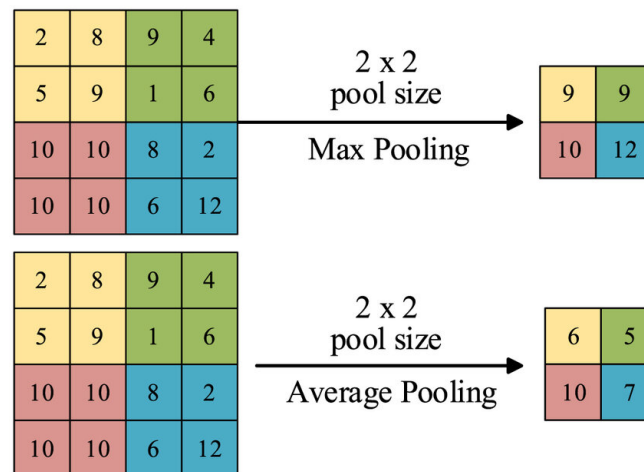


Figura 9.14.: Operaciones de max pooling y average pooling aplicadas a una determinada matriz. Imagen obtenida de [45].

funcionamiento de la red, mientras que las capas anteriores operan independientemente de las dimensiones del volumen de entrada.

En el Capítulo 10, se profundizará en las CNN, introduciendo terminología clave y proporcionando una descripción detallada del proceso de entrenamiento específico para una CNN. Además, se explorarán técnicas avanzadas para optimizar dicho entrenamiento, mejorando así su eficacia y eficiencia

Capítulo 10.

Fundamentos Teóricos

En el capítulo anterior, exploramos diversos campos actuales de la IA, centrándonos específicamente en el DL y culminando en las CNN, la arquitectura central de nuestro estudio.

En este capítulo, se introducen los fundamentos teóricos específicos a esta arquitectura de CNN. Comenzamos definiendo terminología clave y avanzamos hacia una descripción exhaustiva del proceso de entrenamiento exclusivo para las CNN. Se discuten en detalle técnicas avanzadas para optimizar este entrenamiento, tales como ajustes de parámetros, uso de regularizaciones y técnicas de aumento de datos, todo orientado a mejorar la eficacia y eficiencia de las redes.

10.1. Conceptos Clave en el Entrenamiento de Redes

- **Batch.** Un batch o lote es un subconjunto de datos seleccionados del conjunto de entrenamiento completo. Cada batch es utilizado para calcular el gradiente de la función de pérdida y actualizar los parámetros del modelo en una iteración del algoritmo de entrenamiento. El tamaño del batch, que puede variar desde uno hasta el número total de muestras, influye directamente en la velocidad y estabilidad del proceso de entrenamiento. Existen variantes del tradicional algoritmo del Gradiente Descendente [46] que en lugar de utilizar el gradiente completo, calculado a partir de todos los datos disponibles, emplean una estimación del gradiente obtenida a partir del batch [47].
 - **Descenso de Gradiente (por lotes).** El tamaño de cada batch coincide con el conjunto de entrenamiento. Este es en realidad el algoritmo de Descenso del Gradiente original, presentado en la Subsección 9.1.2.
 - **Descenso de Gradiente Estocástico.** El tamaño de cada batch es uno. Con la ayuda de una tasa de aprendizaje que disminuye gradualmente, el Descenso de Gradiente Estocástico tiene la misma tendencia de convergencia que el descenso de gradiente por lotes
 - **Descenso de Gradiente por mini lotes.** El tamaño de cada batch es mayor que uno y menor que el conjunto de entrenamiento. El tamaño del lote es un hiperparámetro (que debe ser ajustado) y los valores comúnmente utilizados son 32, 64, 128, 256, 512. En la literatura puede denominarse también Descenso de Gradiente Estocástico y suele ser el más utilizado.
- **Época.** Cada vez que la red, durante el entrenamiento, visualiza el conjunto de entrenamiento completo.

- **Iteración.** Número de lotes necesarios para completar una época. Durante una iteración, se procesa un solo lote para calcular el gradiente de la función de pérdida y ajustar los parámetros del modelo. Este proceso se repite hasta que se han utilizado todos los lotes del conjunto de entrenamiento, completando así una época.

10.2. Protocolos de Validación Experimental

Existen diversos métodos de protocolos de validación experimental que se utilizan para asegurar la robustez y la eficacia de los modelos. Estos protocolos son fundamentales para evaluar cómo los modelos se comportarán con datos no vistos, un aspecto crítico para su aplicación en entornos reales.

- **Hold-out.** El método de hold-out implica dividir el conjunto de datos en dos segmentos: uno para el entrenamiento y otro como conjunto de test. Este método es sencillo y rápido de implementar, pero su principal desventaja es que el rendimiento del modelo puede depender significativamente de cómo se dividen los datos.
- **Cross-Validation.** En este enfoque, el conjunto de datos se divide en conjunto de entrenamiento y test. Posteriormente el conjunto de entrenamiento se divide en 'k' subconjuntos o pliegues. El modelo se entrena 'k' veces, cada vez utilizando un pliegue diferente como conjunto de validación y los restantes 'k-1' pliegues como conjunto de entrenamiento. Esto permite que cada muestra en el conjunto de datos sea utilizada tanto para entrenamiento como para validación exactamente una vez, maximizando el uso de los datos. Los resultados de las 'k' iteraciones se promedian para obtener una estimación más precisa del rendimiento del modelo
- **Leave-one-out.** Es un caso especial de la validación cruzada donde el número de pliegues 'k' es igual al número de muestras en el conjunto de datos. Este método, es computacionalmente costoso y puede no ser práctico con conjuntos de datos grandes.
- **Único conjunto de Validación** Se trata de una variante del método Hold-Out, que consiste en dividir los datos en un conjunto de prueba y otro de entrenamiento. Sin embargo, la diferencia principal radica en que dentro del conjunto de entrenamiento se reserva un subconjunto adicional para la validación. Este conjunto de validación se emplea para evaluar el rendimiento del modelo sin afectar la independencia del conjunto de prueba. Este enfoque es el más comúnmente utilizado en DL. Se puede visualizar la partición en la Figura 10.1.

10.2.1. Entrenamiento de una CNN

A continuación, se explora el proceso de entrenamiento de una CNN. Durante este proceso, cada capa convolucional ejecuta una serie de tareas específicas necesarias para realizar la actualización de pesos. Aunque este procedimiento ya fue introducido en la Sección 9.3.2, en esta se describe específicamente el proceso de Backpropagation para el caso de una capa convolucional en la que por tanto la operación que se realiza es la convolución. Sea X la entrada, W el filtro (pesos), y la salida y L la función pérdida. Las siguientes operaciones son necesarias:

- **Paso hacia delante (Tarea 1).** La convolución entre la entrada X y el filtro W ,

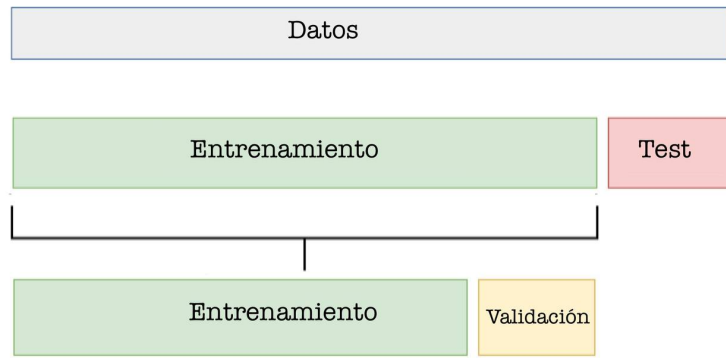


Figura 10.1.: Partición de datos en conjunto de entrenamiento, test y valid.

proporciona una salida y . Esto se puede representar como:

$$y = X * w.$$

- **Paso hacia atrás (Tarea 2).** Los gradientes con respecto a las entradas se calculan convolucionando el núcleo de peso transpuesto con los gradientes con respecto a las salidas. Estos son necesarios para computar el gradiente en la capa anterior.

$$\frac{\partial L}{\partial X} = \frac{\partial L}{\partial y} * W^T.$$

- **Paso hacia atrás (Tarea 3).** Finalmente, los gradientes de la pérdida con respecto al peso se calculan convolucionando cada mapa de características de entrada con los gradientes con respecto a las salidas. Estos son necesarios para realizar la actualización de los pesos W .

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial y} * X.$$

Véase la Figura 10.2 para una representación más visual. Estas tareas, acompañadas de la expresión del gradiente correspondiente a cada una, se utilizan en el artículo de referencia [10] para evaluar posteriormente la eficiencia del modelo presentado, el cual modifica la capa convolucional.

10.3. Técnicas para Mejorar el Entrenamiento

Existen dos fenómenos conocidos como **infraajuste** (underfitting) y **sobreaajuste** (overfitting) [26] que pueden ocurrir al entrenar un modelo de AA. Véase la Figura 10.3

- **Infraajuste.** Ocurre cuando un modelo es demasiado simple, incapaz de capturar la estructura subyacente de los datos. En consecuencia, el modelo no aprende suficientemente bien incluso de los datos de entrenamiento, lo que resulta en un rendimiento pobre.

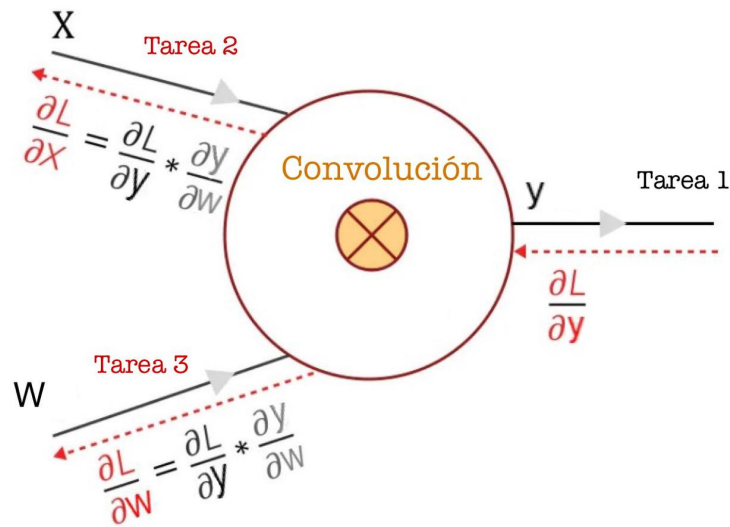


Figura 10.2.: Imagen modificada de [48].

- **Sobreajuste.** Ocurre cuando un modelo se ajusta tan bien a los datos de entrenamiento que se vuelve incapaz de realizar predicciones precisas sobre datos que no ha visto antes, es decir que no es capaz de generalizar. Suele ir asociado a un modelo de una mayor complejidad.

En redes profundas donde están involucrados un gran número de parámetros y se requiere un gran conjunto de datos para realizar el entrenamiento, existe un alto riesgo de que se produzca sobreajuste. A continuación, se explican algunas técnicas empleadas en el entrenamiento de CNN que pueden prevenir la aparición de este fenómeno y mejorar el entrenamiento. Estas técnicas se han estudiado en la asignatura de VC.

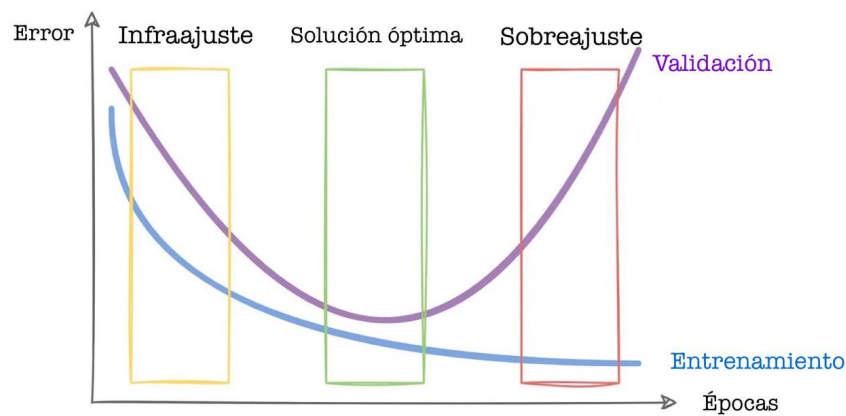


Figura 10.3.: Imagen modificada de [49].

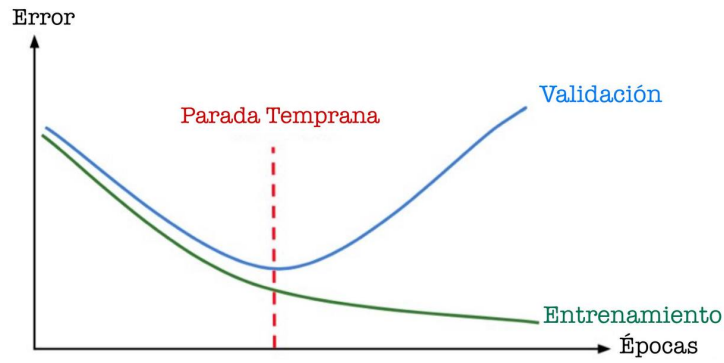


Figura 10.4.: Fenómeno de parada temprana. Imagen modificada de [50].

10.3.0.1. Parada Temprana

Es una técnica de regularización utilizada en el entrenamiento de modelos de AA para evitar el sobreajuste. Funciona deteniendo el proceso de entrenamiento antes de que este haya concluido si no se observa una mejora significativa en el rendimiento del modelo sobre un conjunto de datos de validación. Este rendimiento se evalúa utilizando determinadas métricas. Véase la Figura 10.4.

10.3.0.2. Penalización en la Función de Pérdida

Esta técnica modifica la función de pérdida que se minimiza durante el entrenamiento al agregar un término adicional que penaliza los pesos de gran magnitud, ya que el sobreajuste se asocia con pesos excesivamente grandes. Así, se busca un equilibrio entre minimizar el error del modelo y mantener los parámetros a una magnitud reducida. Algunas de las regularizaciones más conocidas son:

$$\text{Regularización L1: } L(w) = J(w) + \lambda \sum_{i=1}^n |w_i|.$$

$$\text{Regularización L2: } L(w) = J(w) + \lambda \sum_{i=1}^n w_i^2.$$

Donde J es la función de pérdida original y λ es el parámetro de regularización que controla la magnitud de la penalización, un valor más grande restringirá en mayor medida la complejidad del modelo, y viceversa. Por último w_i son los parámetros que se deben aprender.

10.3.0.3. Aumento de Datos

Esta técnica consiste en generar datos adicionales a partir de los datos existentes mediante transformaciones como rotaciones, traslaciones, y cambios en la escala o el color de las imágenes. Esto puede ayudar a mejorar la robustez y generalización del modelo. Dependiendo del problema algunas transformaciones serán válidas y otras no. Por ejemplo en el conjunto de datos de

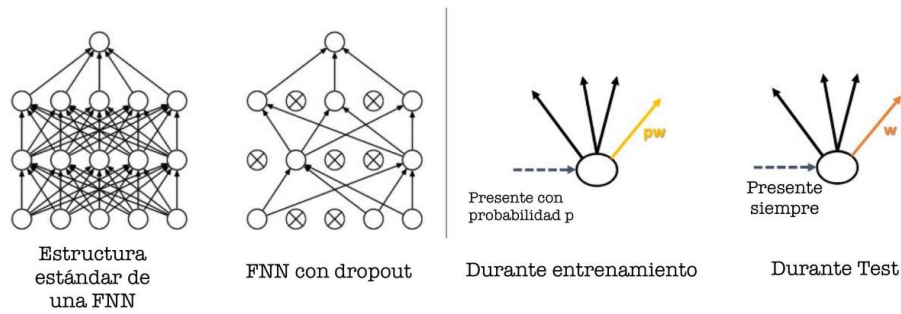


Figura 10.5.: Imagen obtenida de [49].

reconocimiento de dígitos, la rotación de 180° no sería válida ya que los “6” se confundirían con el “9” y viceversa.

10.3.0.4. Batch Normalization

Es habitual en las CNN implementar la Normalización por Lotes (Batch Normalization). Este proceso consiste en normalizar los datos que ingresan a cada capa convolucional con el objetivo de que las operaciones en cada una sean independientes entre sí. Es decir, se busca que la distribución de los datos de entrada a una capa no se vea afectada por los parámetros aprendidos en la capa anterior. Además, este método contribuye a prevenir el sobreajuste de la red, actuando como un mecanismo de regularización. Esta normalización se suele aplicar antes de la capa de activación. Una capa de normalización por lotes también tiene sus propios parámetros:

- Dos parámetros aprendibles β y γ .
- Dos parámetros no aprendibles (Media Móvil de la Media y Media Móvil de la Varianza) que se guardan como parte del ‘estado’ de la capa de normalización por lotes.

Estos parámetros son específicos para cada capa de normalización por lotes.

10.3.0.5. DropOut

Durante el entrenamiento, se desactivan aleatoriamente neuronas ocultas, promoviendo así la generalización de estas al desvincularlas unas de otras. Esta técnica se implementa típicamente en capas completamente conectadas y varían las neuronas afectadas en cada iteración. La “probabilidad de dropout” p es un hiperparámetro clave y se refiere a la probabilidad de que cada neurona sea eliminada durante una iteración específica del entrenamiento. Es importante destacar que el dropout solo se aplica durante el entrenamiento. Para compensar el hecho de que más neuronas están activas en el test y validación, en estos conjuntos se deben utilizar todos los nodos y multiplicar los pesos por el valor de p ($w \times p$) que hemos decidido. Para visualizar el proceso, véase la Figura 10.5.

10.3.0.6. Optimizador Adam

El proceso de aprendizaje en las CNN se lleva a cabo de forma similar a las redes neuronales clásicas, utilizando la técnica de Backpropagation. No obstante, el optimizador de Gradiente Descendente, descrito en secciones anteriores, aplica una tasa de aprendizaje única para todos los pesos de la red y esta tasa permanece constante durante todo el entrenamiento. Por esta razón, se prefiere generalmente el uso del optimizador Adam [51], que no solo permite asignar una tasa de aprendizaje distinta para cada parámetro, sino que también ajusta estas tasas de manera adaptativa.

10.3.1. Data Snooping

El fenómeno de Data Snooping [52] ocurre cuando se utilizan datos del conjunto de prueba, incluso de forma muy sutil. Es crucial evitar estas prácticas ya que el objetivo es resolver un problema específico, no sobreajustar un conjunto de datos en particular. Las decisiones deben basarse en los resultados obtenidos durante las fases de entrenamiento y validación, y no en la fase de prueba.

Capítulo 11.

Transformada de Fourier Discreta

En este capítulo, se analizará la DFT, una herramienta crucial en el procesamiento digital de señales. Se detallarán algunas de sus propiedades más importantes que resultarán familiares y vendrán motivadas por su similitud con las propiedades descritas en el ámbito continuo. Adicionalmente, se examinará cómo la transformación al dominio de Fourier puede simplificar y clarificar diversas operaciones en el filtrado de imágenes, concretamente la operación de convolución, que como se describió en el Capítulo 9, es el eje fundamental en la arquitectura de las CNN. Por ello, el capítulo profundiza en el estudio de esta operación y culmina con el Teorema de Convolución en el marco discreto, que como se describió con anterioridad, es la pieza clave para el método alternativo a la convolución tradicional que se busca aprovechar en el entrenamiento de CNN.

11.1. Motivación

El ejemplo más común y conocido del contenido frecuencial en las señales probablemente sean las señales de audio, especialmente la música. Estamos familiarizados con las notas musicales denominadas “altas” y “bajas”, y con la expresión general de un senoide a una frecuencia ω (o frecuencia f en Hertz), fase ϕ y amplitud a :

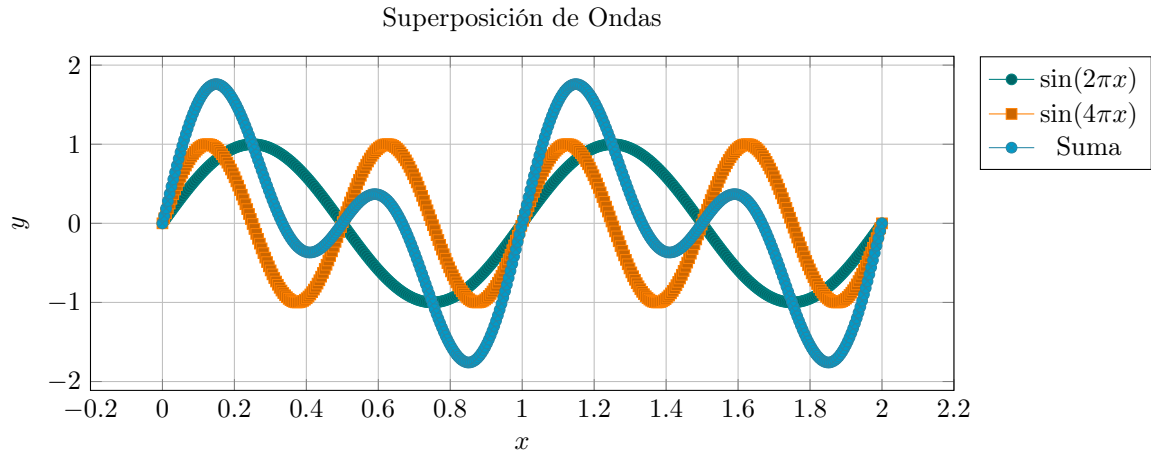
$$x(t) = a \sin(\omega t + \phi) = a \sin(2\pi f t + \phi)$$

La amplitud revela la intensidad de una frecuencia específica dentro de una señal, mientras que la fase señala la posición de esa frecuencia en el contexto temporal de la señal. Considerado como una señal de audio, $x(t)$ indica los cambios en la presión del aire sobre nuestros oídos en función del tiempo. Podemos combinar dos sinusoides sumando las señales de la manera habitual. Por ejemplo,

$$x(t) = \sin(2\pi t) + \sin(4\pi t), \tag{11.1}$$

que es una combinación de un senoide con frecuencia de 1 Hz y un senoide con frecuencia de 2 Hz. En la ecuación (11.1), la amplitud de cada senoide es $a = 1$ y la fase de cada uno es $\phi = 0$. Una gráfica de $x(t)$ se muestra en la Figura 11.1. El “sonido” creado por $x(t)$ es la combinación de los dos tonos que constituyen $x(t)$.

Así, al combinar señales simples se generan señales más complejas. De la misma manera se querría realizar el proceso inverso y poder expresar una señal compleja como composición de señales más sencillas. De esta forma se podría simplificar su análisis y facilitar el filtrado y

Figura 11.1.: Gráfica de $\sin(2\pi x)$, $\sin(4\pi x)$ y su suma.

procesamiento de la señal, donde componentes particulares puedan ser modificadas o eliminadas sin afectar el conjunto global. La DFT en una dimensión (DFT-1D) permitirá expresar una señal unidimensional compleja en términos de señales más sencillas, y la inversa de la DFT permitirá recuperar exactamente la señal original a partir de estas componentes simplificadas. Si la señal es continua, se debe realizar un muestreo previo para establecer un conjunto de valores finitos a los que se calculará la DFT-1D.

Es importante destacar que la calidad del muestreo afecta significativamente la calidad de la señal transformada. El Teorema de Muestreo de Nyquist-Shannon [53] dicta que la frecuencia de muestreo debe ser al menos el doble de la frecuencia más alta en la señal para capturar toda la información sin incurrir en *aliasing* (distorsión de la señal debida a un muestreo inadecuado) [54]. Si el muestreo no se realiza correctamente, se pueden introducir errores que afectarán los resultados de la DFT, y puede ocurrir que no se recupere correctamente la señal realizando la inversa.

Por otra parte, una imagen es esencialmente una señal visual en 2-D de naturaleza discreta, por lo tanto es razonable esperar que se pueda descomponer de la misma manera que las señales en 1-D. Para lograr esto, se emplea la DFT en dos dimensiones (DFT-2D), que permite analizar y manipular las características frecuenciales de la imagen de manera similar a como se hace con señales unidimensionales. De manera que, por ejemplo, se podrían distinguir las frecuencias bajas, que representan la información más general, de las frecuencias altas, que detallan los aspectos más finos.

Esto sería útil en muchos ámbitos ya que se podría almacenar el grado de detalle que se necesite. Por ejemplo, un proceso que consumiría mucho tiempo sería navegar por una amplia base de datos, buscar un tipo específico de imágenes, descargar cada imagen completamente, y terminar determinando que la mayoría no cumple con lo buscado. Ciertamente, sería posible descartar la mayoría de las imágenes con solo una idea muy básica de su contenido. Lo ideal sería tomar esta decisión con solo un determinado porcentaje de los datos (por ejemplo con la mitad). De esta forma, la descarga y revisión de la base de datos sería mucho más rápida para localizar lo

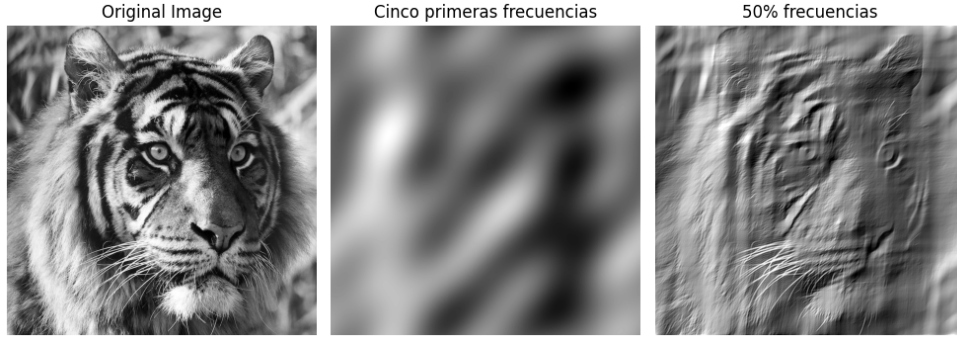


Figura 11.2.: De izquierda a derecha se muestran la imagen original, las cinco primeras frecuencias y el 50 % de las frecuencias de la misma.

que se busca.

El problema es que si se envía la primera mitad de los píxeles, se obtendría solo una fracción (específicamente $1/2$) de la imagen original, aunque a alta resolución. Una estrategia más eficaz sería separar la información de la imagen en componentes de detalle “grueso” y “fino”. Este enfoque permite una representación jerárquica de la imagen, donde se transmite primero la información más esencial o básica, y luego, según sea necesario, se añaden detalles adicionales. Esta metodología es particularmente útil en situaciones donde la banda ancha es limitada o donde el usuario necesita una vista previa rápida antes de decidir si requiere o no la imagen completa con todos sus detalles.

La Figura 11.2 ilustra cómo se puede representar una imagen utilizando diferentes porcentajes de su información frecuencial. La compresión de imágenes moderna explota esta idea al permitir una carga incremental de la calidad de la imagen.

11.2. Transformada de Fourier Discreta 1D

En la primera parte del trabajo se estudió la Transformada de Fourier en $\mathbb{R}^n$. En particular, la definición para $n = 1$ es la siguiente:

Sea $f : \mathbb{R} \rightarrow \mathbb{C}$ integrable en $\mathbb{R}$, dado $y \in \mathbb{R}$:

$$\hat{f}(y) = \int_{\mathbb{R}} f(x) e^{-2\pi i x y} dx.$$

La definición para la DFT [1] debería ir en consonancia con la anterior. No obstante, en este caso la variable x no puede variar a lo largo de todo $\mathbb{R}$, sino que ahora se limitará a un número finito de valores, correspondiente al número de muestras equiespaciadas que se recojan de la señal original.

Definición 11.2.1. Sea $f = (f[0], f[1], \dots, f[N-1])$ una N-tupla. Se define la DFT-1D de f

como la N-tupla $\hat{f}$ tal que:

$$\hat{f}[m] = \sum_{k=0}^{N-1} f[k] e^{-2\pi i \frac{mk}{N}}, \quad m = 0, 1, \dots, N-1. \quad (11.2)$$

Usando la Fórmula de Euler, se tiene que

$$\hat{f}[m] = \sum_{k=0}^{N-1} f[k] \left(\cos\left(-2\pi \frac{mk}{N}\right) + i \sin\left(-2\pi \frac{mk}{N}\right) \right), \quad m = 0, 1, \dots, N-1.$$

Fijado $m \in \{0, \dots, N-1\}$, el término $\hat{f}[m]$ en la ecuación de la DFT representa la suma ponderada de todas las muestras $f[k]$, donde cada muestra está multiplicada por un factor oscilante que representa una combinación de funciones seno y coseno. La señal original queda por tanto representada por este vector $\hat{f}$ de números complejos.

Para cada número complejo $\hat{f}[m]$, se puede calcular la magnitud A_m y la fase ϕ_m :

$$A_m = \frac{|\hat{f}[m]|}{N} = \frac{\sqrt{\text{Re}(\hat{f}[m])^2 + \text{Im}(\hat{f}[m])^2}}{N},$$

$$\phi_m = \text{atan2}(\text{Im}(\hat{f}[m]), \text{Re}(\hat{f}[m])),$$

donde $\text{Im}(\hat{f}[m])$ y $\text{Re}(\hat{f}[m])$ son, respectivamente, las partes imaginaria y real del número complejo $\hat{f}[m]$, y atan2 es la función arco tangente de dos argumentos.

La **magnitud** de cada uno de estos números complejos indica cuánta presencia tiene esa frecuencia particular en la señal original. Por otro lado, la **fase** de cada componente frecuencial indica el desplazamiento temporal o espacial del senoide básico asociado a esa frecuencia.

En el dominio del tiempo, una señal unidimensional se visualiza y se entiende como una serie de valores o intensidades a lo largo del tiempo. Por ejemplo, en el caso de una señal de audio, estos valores representan variaciones de presión del aire en el tiempo. Al calcular la DFT-1D se efectúa un cambio fundamental ya que la señal se describe en términos de un vector de números complejos $\hat{f}$ que representan las amplitudes y las fases de sus componentes de frecuencia. Por ello, se dice que la DFT-1D transita del **dominio del tiempo** al **dominio de la frecuencia**.

En la primera parte del TFG se estudió la Transformada Inversa de Fourier en $\mathbb{R}^n$. Esta difería de la Transformada en el signo que aparecía en la exponencial. Del mismo modo se tiene ahora la siguiente definición.

Definición 11.2.2. Sea $F = (F[0], F[1], \dots, F[N-1])$ una N-tupla. Se define la Inversa de la DFT-1D (IDFT-1D) de F como la N-tupla f tal que:

$$f[n] = \frac{1}{N} \sum_{m=0}^{N-1} F[m] e^{2\pi i \frac{mn}{N}}, \quad n = 0, 1, \dots, N-1. \quad (11.3)$$

Es sencillo probar que la composición de (11.2.1) y (11.2.2) produce la identidad. Dichas ecuaciones definen un par de DFT. Adicionalmente, indican que la DFT e IDFT existen para

cualquier conjunto de muestras cuyos valores sean finitos. La DFT define una biyección lineal en $\mathbb{C}$.

Observación 11.2.3. Se suele adoptar el siguiente convenio: Al operar con la DFT de una secuencia de N elementos $\{(f[0], f[1], \dots, f[N-1])\}$, se considera que dicho vector es una muestra de una sucesión infinita periódica con período N . Por lo que para cualquier entero k arbitrario, $f[k]$ se define como $f[q]$, donde $0 \leq q \leq N-1$ es el resto de la división de k por N .

Teorema 11.2.4. *La DFT 1-D transforma señales periódicas discretas en el dominio del tiempo en señales periódicas discretas en el dominio de la frecuencia. Análogamente, la IDFT transforma señales periódicas discretas en el dominio de la frecuencia en señales periódicas discretas en el dominio del tiempo.*

Demostración. La prueba de la periodicidad de la DFT se deduce directamente a partir de la definición.

$$\widehat{f}[m+N] \stackrel{\text{def}}{=} \sum_{k=0}^{N-1} f[k] e^{-\frac{2\pi i}{N}(m+N)k} = \sum_{k=0}^{N-1} f[k] e^{-\frac{2\pi i}{N}mk} \underbrace{e^{-2\pi i k}}_1 = \sum_{k=0}^{N-1} f[k] e^{-\frac{2\pi i}{N}mk} = \widehat{f}[m].$$

De manera equivalente, se obtiene la prueba de la periodicidad de la IDFT-1D. $\square$

A continuación se presenta la DFT en dos dimensiones, que es la que típicamente se aplica al contexto de imágenes, aunque esta se puede descomponer en sucesivas transformadas unidimensionales.

11.3. Transformada de Fourier Discreta 2D

Una imagen en escala de grises se puede describir como una función $I : \mathbb{R}^2 \rightarrow \mathbb{R}$, donde $I(x, y)$ asigna un valor de intensidad a cada punto (x, y) . Dado que esta función es intrínsecamente discreta, surge la necesidad de adaptar la definición de la DFT para abordar el caso bidimensional. Para ello se puede atender a la expresión de la Transformada de Fourier en $\mathbb{R}^2$, descrita en la parte matemática de la memoria, concretamente el Capítulo 4.

Sea $f : \mathbb{R}^2 \rightarrow \mathbb{C}$ integrable en $\mathbb{R}^2$, dado $(y_1, y_2) \in \mathbb{R}^2$:

$$\widehat{f}(y_1, y_2) = \int_{\mathbb{R}^2} f(x_1, x_2) e^{-2\pi i(x_1 y_1 + x_2 y_2)} dx_1 dx_2 = \int_{\mathbb{R}} e^{-2\pi i(x_2 y_2)} \int_{\mathbb{R}} f(x_1, x_2) e^{-2\pi i(x_1 y_1)} dx_1 dx_2.$$

Si se trata de concebir esta expresión de manera discreta, se tiene, que para un valor concreto de x_2 , se mueve x_1 a lo largo de todo el dominio considerado, calculando $f(x_1, x_2) e^{-2\pi i(x_1 y_1)}$ y acumulando los resultados obtenidos. Este proceso se repite con todos los valores de x_2 en el dominio correspondiente.

En efecto, esto nos lleva a la definición de la DFT-2D [1]:

Definición 11.3.1. Sea f una matriz de tamaño $N \times M$. Se define la DFT-2D de f como la matriz $\widehat{f}$ definida por:

$$\widehat{f}[k, l] = \sum_{n=0}^{N-1} \sum_{m=0}^{M-1} f[n, m] e^{-2\pi i \left(\frac{kn}{N} + \frac{lm}{M} \right)}, \quad k = 0, 1, \dots, N-1 \quad \text{y} \quad l = 0, 1, \dots, M-1. \quad (11.4)$$

De manera análoga al caso unidimensional, cuando se trabaja con imágenes en el dominio espacial, la transformación de la DFT-2D resulta en una nueva matriz compuesta por números complejos. Cada uno de estos números se puede caracterizar por su magnitud y fase, elementos que permiten representar la imagen en términos de frecuencias.

Para cada número complejo $\widehat{f}[k, l]$, se puede calcular la magnitud $A_{k,l}$ y la fase $\phi_{k,l}$:

$$A_{k,l} = \frac{|\widehat{f}[k, l]|}{N} = \frac{\sqrt{\text{Re}(\widehat{f}[k, l])^2 + \text{Im}(\widehat{f}[k, l])^2}}{N},$$

$$\phi_{k,l} = \text{atan2}(\text{Im}\widehat{f}[k, l], \text{Re}(\widehat{f}[k, l])).$$

Por esta razón, se afirma que la DFT-2D convierte la imagen del **dominio espacial** al **dominio de la frecuencia**.

Definición 11.3.2. Sea F una matriz de tamaño $N \times M$. Se define la Inversa de la DFT bidimensional (IDFT-2D) de f como la matriz f definida por:

$$f[n, m] = \frac{1}{NM} \sum_{k=0}^{N-1} \sum_{l=0}^{M-1} F[k, l] e^{2\pi i \left(\frac{kn}{N} + \frac{lm}{M} \right)}, \quad n = 0, 1, \dots, N-1 \quad \text{y} \quad m = 0, 1, \dots, M-1. \quad (11.5)$$

Es claro que la composición de (11.3.1) y (11.3.2) constituye la identidad. Ambas constituyen un par de DFT-2D.

Es interesante observar la Figura 11.3, donde se muestra la reconstrucción de una imagen utilizando únicamente la información de la magnitud (normalizando la fase a 0) y únicamente la información de la fase (normalizando la magnitud a 1). Esta ilustración demuestra que, contrariamente a lo que se podría suponer, la fase contiene más información representativa de la imagen, permitiendo reconocerla a través de ella, mientras que la magnitud sola hace imposible esta tarea. Por lo tanto, si se tuviera que elegir preservar una de las dos, esta sería la fase.

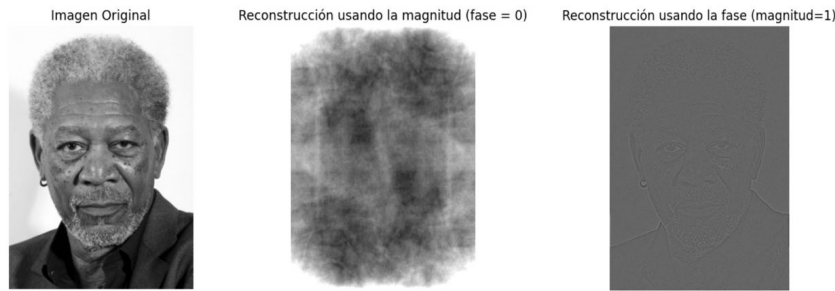


Figura 11.3.: De izquierda a derecha se muestra: la imagen original, la reconstrucción de la misma usando únicamente la magnitud, y la reconstrucción usando únicamente la fase.

Obsérvese a continuación la Figura 11.4, la cual ilustra un ejemplo de cómo se puede modificar una imagen mediante la manipulación de su representación en el dominio de la frecuencia. El proceso comienza con el cálculo de la DFT, obteniendo una representación de la imagen en el dominio de la frecuencia. En la Transformada, se detecta un punto particular centrado en la imagen, denominado punto DC (Direct Current) y que corresponde con la frecuencia cero dentro del espectro. Matemáticamente, representa la suma total de todos los valores de los píxeles de la imagen, dividido por el número de píxeles, es decir, el promedio de los valores de los píxeles.

El componente DC es crucial para entender el nivel general de intensidad de una imagen. En realidad, al aplicar la DFT, el componente DC aparece en la esquina superior izquierda del espectro de frecuencias. Sin embargo, para un análisis más intuitivo y una mejor visualización, es más útil tener las frecuencias bajas agrupadas en el centro de la imagen Transformada y las altas hacia los bordes.

Con esta representación espectral se puede manipular la imagen Transformada para ajustar su contenido de frecuencia espacial:

- Inicialmente, se busca aplicar un efecto de suavizado en la imagen; es decir, se requiere aplicar una especie de filtro de paso bajo que permita pasar los componentes de baja frecuencia espacial, pero corte las frecuencias espaciales altas. Dado que los componentes de baja frecuencia se encuentran cerca de DC, se define un radio alrededor del punto DC y se pone a cero cada punto en la imagen Transformada que esté más allá de ese radio. La aplicación de la IDFT a esta imagen filtrada de paso bajo produce la imagen mostrada en la segunda columna de la segunda fila de la Figura 11.4. El resultado es una imagen suavizada, donde se preservan las áreas de luz y sombra amplias y suaves, mientras que se pierden los detalles finos y los bordes nítidos, proporcionando un efecto visual general de desenfoque.
- A continuación, se prueba lo contrario, el filtrado de paso alto, donde usamos el mismo umbral de frecuencia espacial para definir un radio en la imagen transformada. Todos los componentes de frecuencia espacial que caen dentro de ese radio se eliminan, preservando solo los componentes de alta frecuencia. Después de realizar la IDFT, se comprueba el efecto del filtrado de paso alto, que consiste en conservar todos los bordes nítidos y

definidos del original, pero se pierden las regiones más grandes de oscuro y claro.

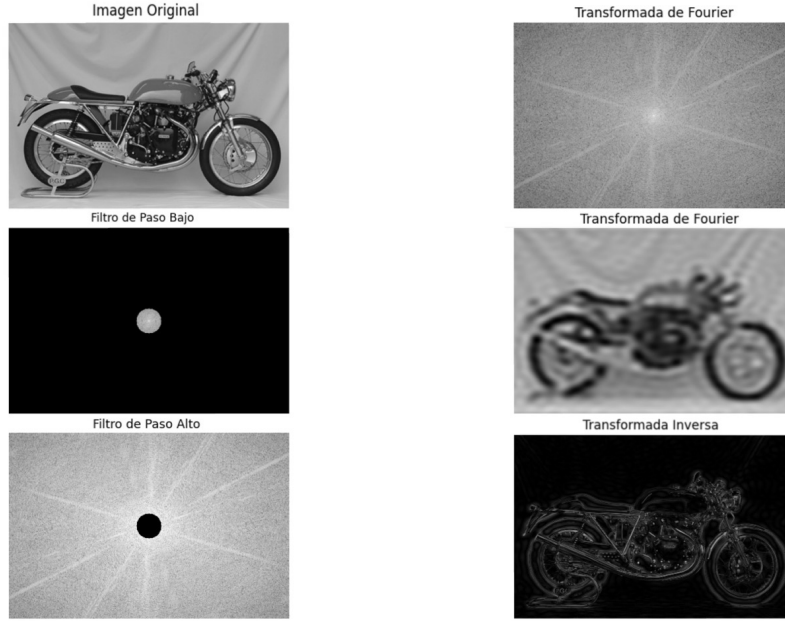


Figura 11.4.: En la primera fila se muestra una imagen junto con su correspondiente DFT, en las siguientes dos filas se aplica un filtrado de paso bajo y paso alto, respectivamente, con el procedimiento descrito anteriormente.

Teorema 11.3.3. *La DFT 2-D puede descomponerse en la aplicación secuencial de DFT 1-D. Esta propiedad es conocida como separabilidad [1].*

Demostración. Dado $k \in \{0, 1, \dots, N-1\}$ y $l \in \{0, 1, \dots, M-1\}$, veamos que podemos escribir la DFT como

$$\hat{f}[k, l] = \sum_{n=0}^{N-1} e^{-2\pi i \left(\frac{kn}{N}\right)} \sum_{m=0}^{M-1} f[n, m] e^{-2\pi i \left(\frac{lm}{M}\right)} = \sum_{n=0}^{N-1} e^{-2\pi i \left(\frac{kn}{N}\right)} \hat{g}[n, l], \quad (11.6)$$

donde, dado $n \in \{0, 1, \dots, N-1\}$,

$$\hat{g}[n, l] = \sum_{m=0}^{M-1} f[n, m] e^{-2\pi i \left(\frac{lm}{M}\right)}. \quad (11.7)$$

Para un valor de n , se tiene que $\hat{g}(n, l)$ es la DFT 1-D de la n -ésima fila de $f(x, y)$. Al variar n desde 0 hasta $M-1$ en la ecuación (11.3), se calcula un conjunto de DFT 1-D para todas las filas de $f(x, y)$. Posteriormente, se le realizan transformaciones 1-D a las columnas de este conjunto resultante $\hat{g}(n, l)$. Así, se concluye que la DFT 2-D de $f(x, y)$ puede obtenerse mediante el cálculo de la transformada 1-D de cada fila de $f(x, y)$ seguido de calcular la DFT 1-D a lo largo de cada columna del resultado. Esta es una simplificación importante que permite lidiar

con una variable a la vez. □

Observación 11.3.4. De hecho, las librerías que calculan la DFT de una determinada señal 2D, en realidad aplican sucesivas transformadas 1D y usan el algoritmo FFT 1-D (usando el algoritmo FFT descrito en el siguiente capítulo).

11.4. Filtrado Lineal

Como se ha mencionado anteriormente, una imagen en escala de grises puede ser representada como una función. En el caso de imágenes a color ocurre lo mismo.

Dado que una imagen puede ser tratada como una función, es posible aplicarle diversos operadores que pueden ser globales o locales. Sería inapropiado aplicar operadores a píxeles individuales de la imagen, dado que lo relevante no son los píxeles por sí mismos, sino las relaciones que mantienen entre sí. La Figura 11.5 demuestra claramente este concepto.



Figura 11.5.: Ejemplos de imágenes con la misma cantidad de información.

Un ejemplo de operadores globales se ilustra en la Figura 11.6. Sin embargo, suele ocurrir que, mientras que los píxeles distantes entre sí siguen un comportamiento que puede variar considerablemente, los píxeles más cercanos muestran cierta dependencia entre sus valores. Por ello, existen otras operaciones, como el filtrado, que implican modificar la imagen al aplicar una función (local) específica en el entorno de cada píxel. Dentro de estas técnicas se incluye el filtrado lineal [55], el cual consiste en reemplazar cada píxel por una combinación lineal de los valores de sus vecinos, aplicando una suma ponderada. Los pesos para la combinación lineal se denominan filtro o “kernel”.

En esta sección, se estudiarán específicamente los operadores de filtrado lineal denominados **correlación** y **convolución**. Al final, tanto la imagen como el kernel se representan como matrices numéricas y, por lo tanto, asignar a una el nombre de “imagen” y a la otra de “kernel” es principalmente una cuestión de función más que de forma. En el procesamiento de imágenes, esta distinción se basa en el uso que se le da a cada matriz: la “imagen” generalmente se refiere al objeto de análisis o procesamiento, mientras que el “kernel” se aplica sobre la imagen para modificarla o extraer información de ella.

Sin embargo, puede ser relevante el papel asignado a cada elemento si la operación que se realiza no es conmutativa. Se asumirá que el kernel es cuadrado pero cabe destacar que los principios

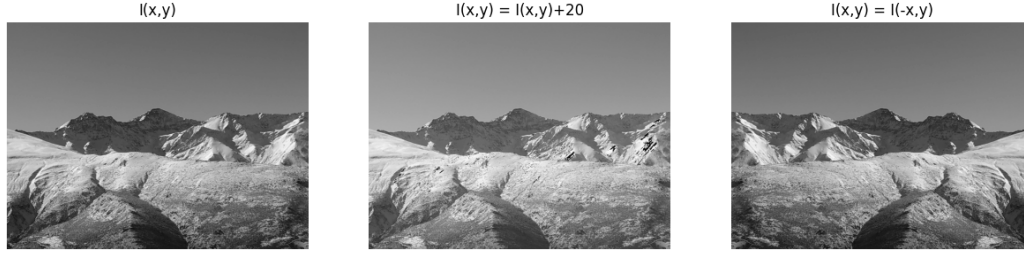


Figura 11.6.: Aplicación de operadores globales a la imagen original $I(x,y)$.

de la convolución aplican igualmente bien a kernels de forma no cuadrada. Las operaciones de convolución y correlación fueron descritas y comparadas en la primera parte de la memoria, concretamente en el Capítulo 5. En esta sección, se realizará un análisis adicional enfocado al contexto específico que se está tratando.

11.4.1. Correlación

Definición 11.4.1. Sea $F = (F[n, m])$ una imagen de tamaño $N \times M$, y K un kernel de tamaño $(2k + 1) \times (2k + 1)$. Se define la operación de correlación entre F y K como

$$(F \star K)[i, j] = \sum_{u=-k}^k \sum_{v=-k}^k K[u, v] F[i + u, j + v], \quad (11.8)$$

para $i \in \{0, \dots, N - 1\}$ y $j \in \{1, \dots, M - 1\}$.

Refiriéndose a la expresión en (11.4.1), el proceso implica que el kernel se desplace sobre la imagen, ocupando cada posición posible. En cada ubicación, se realiza una multiplicación punto a punto entre los valores del kernel y los valores subyacentes de la imagen. Las multiplicaciones resultantes se suman para formar el valor de correlación correspondiente a esa posición específica, el cual se asigna al mismo lugar en la matriz de salida. Este procedimiento se repite para todas las posiciones posibles del kernel sobre la imagen, generando una nueva matriz de salida. Esta matriz actúa como un “mapa” que ilustra el grado de coincidencia o correlación entre el kernel y cada área de la imagen, por lo que es especialmente útil para detectar patrones o realizar comparación de plantillas.

11.4.2. Convolución

Definición 11.4.2. Sea $F = (F[n, m])$ una imagen de tamaño $N \times M$, y K un kernel de tamaño $(2k + 1) \times (2k + 1)$. Se define la operación de convolución entre F y K como

$$(F * K)[i, j] = \sum_{u=-k}^k \sum_{v=-k}^k K[u, v] F[i - u, j - v], \quad (11.9)$$

para $i \in \{0, \dots, N - 1\}$ y $j \in \{1, \dots, M - 1\}$.

Atendiendo a la expresión (11.4.2), se observa su similitud con (11.4.1). De hecho, la operación

de convolución consiste primero en girar el núcleo horizontal y verticalmente de arriba a abajo y de izquierda a derecha (es decir, rotarlo 180°), y posteriormente realizar la operación de correlación. Esta operación, en ocasiones denominada convolución lineal, mide el efecto que ejerce el núcleo sobre la propia imagen.

Observación 11.4.3. Nótese que si el núcleo es simétrico, tanto la operación de convolución como de correlación proporcionan la misma salida.

En el procesamiento de señales y análisis de imágenes, el manejo de los bordes durante la correlación afecta a las dimensiones de la imagen de salida. Existen tres modos principales que determinan estas dimensiones: “full”, “same”, y “valid”. Véase la Figura 11.7.

- **Full:** El modo “full” proporciona la salida más extensa. Aquí, el kernel se desliza sobre cada punto posible de la imagen, incluyendo los bordes y esquinas, resultando en una imagen de salida que es mayor que la imagen original.

Dimensiones de salida: Si la imagen es de tamaño $M \times N$ y el kernel de tamaño $P \times P$, entonces la imagen de salida será de $(M + P - 1) \times (N + P - 1)$.

- **Same:** El modo “same” asegura que la imagen de salida tenga el mismo tamaño que la imagen de entrada. Se añade relleno adecuado a la imagen original para compensar el tamaño del kernel.

Dimensiones de salida: La imagen de salida mantiene las dimensiones originales $M \times N$.

- **Valid:** El modo “valid” sólo considera las posiciones donde el kernel se ajusta completamente dentro de la imagen original. No se agrega relleno, y el kernel se aplica únicamente en las regiones donde puede encajar completamente.

Dimensiones de salida: Si la imagen original es de $M \times N$ y el kernel es de $P \times P$, entonces la imagen de salida será de $(M - P + 1) \times (N - P + 1)$.

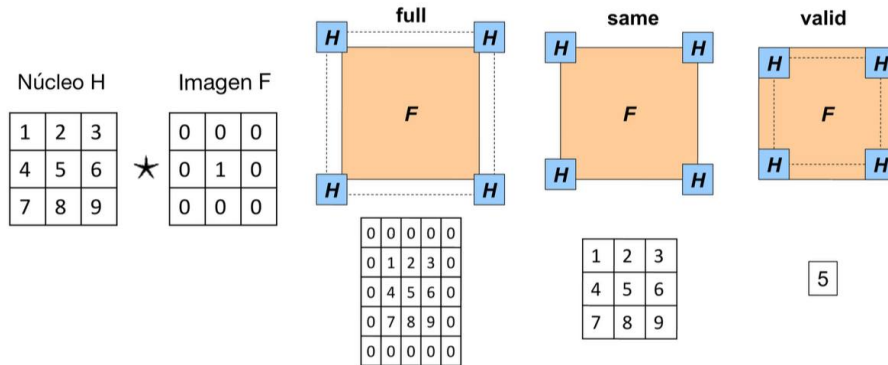


Figura 11.7.: Correlación del núcleo H y la imagen F utilizando los modos full, same y valid. Imagen obtenida de la asignatura VC.

11.4.2.1. Propiedades

En el contexto del AA, el algoritmo de aprendizaje determinará los valores adecuados del núcleo en el lugar apropiado, de modo que un algoritmo basado en la convolución aprenderá un núcleo que estará girado en comparación con el núcleo aprendido por un algoritmo basado en correlación. Sin embargo, existen otros contextos donde las diferencias en el comportamiento de las operaciones son notables a pesar de la sutil diferencia en cada una de sus expresiones matemáticas. De hecho, la convolución posee algunas propiedades que la correlación no tiene y, en cierto sentido, ofrece una respuesta más natural.

- **Linealidad.** Si F y H son imágenes, $a, b \in \mathbb{R}$, y L la operación de convolución o de correlación, entonces

$$L(aF + bH) = aL(F) + bL(H).$$

Esto es debido a que ambas operaciones son de filtrado lineales.

- **Equivariante respecto a traslaciones.** Si F y H son imágenes, T es una traslación y L la operación de convolución o de correlación, entonces

$$L(T(F), H) = T(L(F, H)).$$

De esta manera, si una imagen de entrada se traslada, el resultado tanto de la operación de correlación como de convolución experimentará una traslación equivalente.

- **Conmutatividad.** Si F es una imagen y K es un núcleo, entonces la operación de convolución es conmutativa:

$$F * K = K * F.$$

Esto no ocurre con la operación de correlación.

- **Asociatividad.** Si F es una imagen y K, H son núcleos. Entonces la operación de convolución es asociativa:

$$F * K * H = F * (K * H).$$

Esto no ocurre con la operación de correlación.

- **Identidad.** Si F representa una imagen y K es un núcleo donde todas las entradas son cero excepto la central. Entonces

$$F * K = F.$$

Este resultado es el esperado en términos de identidad, ya que el filtro no altera la imagen. Este comportamiento no se observa con la correlación.

La propiedad de asociatividad permite la pre-convolución de filtros, es decir, la construcción de filtros complejos a partir de filtros simples, de modo que solo sea necesario convolucionar la imagen con un único filtro. Por tanto, si existe una imagen F y se desea convolucionarla primero con K y luego con H , se puede convolucionar K y H previamente para crear un único filtro. Posteriormente, se convoluciona F con este filtro combinado.

Sin embargo, esto no ocurre con la operación de correlación, por lo que si se necesita utilizar múltiples filtros en sucesión y realizar esta operación en varias imágenes, es conveniente usar la operación de convolución para convertir previamente los múltiples filtros en uno único.

Numerosas bibliotecas de AA implementan la correlación cruzada, aunque la denominan con-

volución. Un ejemplo de esto es la función `cv2.filter2D` de OpenCV [56]. Por esta razón, es crucial consultar la documentación correspondiente.

11.4.2.2. Filtro Gaussiano

Dentro de los diversos filtros disponibles para el procesamiento de imágenes, el filtro Gaussiano [57] es particularmente destacado. Conocido también como Gaussian blur, esta técnica es ampliamente utilizada para suavizar imágenes, reduciendo tanto el ruido como los detalles innecesarios. El nombre de este filtro proviene de la función Gaussiana, o campana de Gauss, que se emplea para crear una matriz aplicada posteriormente sobre la imagen, como se explica detalladamente en la Observación 4.7.5. Los dos parámetros principales de este filtro son σ y el tamaño del kernel. Por un lado, σ es la desviación estándar que controla la amplitud de la campana de la función gaussiana, afectando la extensión del suavizado. Por otro lado, el tamaño del kernel determina las dimensiones de la matriz utilizada para la convolución, influyendo directamente en cuántos píxeles vecinos se consideran para cada cálculo y, por ende, en la intensidad del efecto de suavizado. Véase la Figura 11.8.

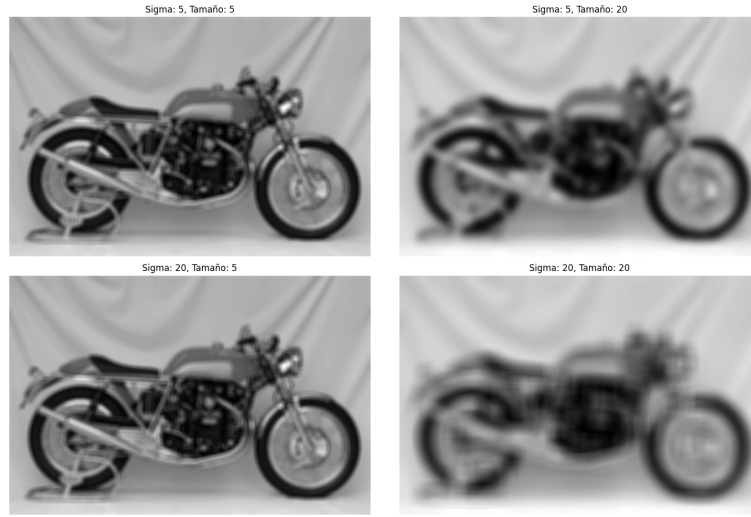


Figura 11.8.: Se presentan diversos resultados de filtrado variando los valores de σ y T.

11.4.3. Convolución Circular

Existe otro tipo de convolución, denominado convolución circular o convolución cíclica, cuya definición se presenta a continuación.

Definición 11.4.4. Sea $F = (F[n, m])$ una imagen de tamaño $N \times M$, y K un kernel de tamaño $(2k + 1) \times (2k + 1)$. La convolución circular entre F y K se define como

$$(F \circledast K)[i, j] = \sum_{u=-k}^k \sum_{v=-k}^k K[u, v] F[(i - u) \bmod N, (j - v) \bmod M],$$

para $i \in \{0, \dots, N - 1\}$ y $j \in \{1, \dots, M - 1\}$.



Figura 11.9.: De izquierda a derecha se presenta: la imagen original, la convolución lineal, y la convolución circular sobre esta.

La expresión para este tipo de convolución es similar a la que se describe en (11.4.2) pero con la diferencia clave de que los datos se tratan de manera periódica. En la convolución circular, los elementos al final de la matriz se conectan con los del principio, creando un efecto de “envoltura” o “wrap around”. Esto significa que cuando un elemento del kernel se superpone más allá de los bordes de la matriz, no se considera como un elemento exterior o nulo, sino que se asume que la matriz es cíclica y los datos continúan desde el lado opuesto.

En la Figura 11.9, se presenta el resultado de aplicar tanto la convolución circular como la lineal sobre una misma imagen a la que se le ha añadido un borde artificial en el lado izquierdo. En la convolución circular, este borde influye en el borde derecho debido al efecto de envoltura, lo que resulta en diferencias notables entre las dos convoluciones.

Este comportamiento es esencial en aplicaciones donde la señal o los datos son inherentemente periódicos, como en el procesamiento de señales digitales o análisis de series temporales cíclicas.

Sin embargo, en el procesamiento de imágenes es más común usar la convolución lineal ya que no se asume ninguna periodicidad en los datos, lo que la hace adecuada para la mayoría de las aplicaciones en imágenes que no son inherentemente cíclicas. De hecho, en el filtrado se utiliza específicamente la convolución lineal cuando no se desea que los datos en un lado de la imagen afecten los datos del otro lado, o si es importante manejar los bordes específicamente (por ejemplo, aplicando diferentes pesos o evitando el efecto de los bordes en el resultado).

Proposición 11.4.5. *Bajo ciertas condiciones la convolución lineal y circular de una imagen I de tamaño $P_1 \times P_2$ y un kernel K de tamaño $Q \times Q$ coinciden:*

1. *Se eligen N_1 y N_2 tales que $N_1 \geq P_1 + Q - 1$ y $N_2 \geq P_2 + Q - 1$.*
2. *Se rellenan con ceros las matrices $DFT(I)$ y $DFT(K)$ de manera uniforme para que ambas tengan un tamaño de $N_1 \times N_2$. El relleno debe aplicarse tanto en los lados superior e inferior como en los lados izquierdo y derecho de las matrices, de manera que se mantengan en el centro las señales.*

11.5. Teorema de Convolución

Finalmente, se explora el Teorema de Convolución, que proporciona una herramienta esencial para realizar convoluciones como productos puntuales. Como ya se viene advirtiendo esta aproximación ofrecerá una posible alternativa al método directo de cálculo de convoluciones.

El Teorema de Convolución en $\mathbb{R}^n$, descrito en la Sección 5.2, establece que la convolución de dos funciones coincide con el producto de las Transformadas de Fourier de ambas.

Sin embargo, en el ámbito discreto, el Teorema de Convolución 2D utiliza la DFT y la IDFT, las cuales generan secuencias de datos periódicas. Por ello, este teorema será equivalente al de $\mathbb{R}^n$ con la salvedad de que la convolución a la que se refiere es a la circular.

Teorema 11.5.1 (de Convolución). *Sea F una imagen y K un núcleo con las consideraciones anteriores. Entonces*

$$(\widehat{F \circledast K})(x, y) = \widehat{F}(x, y) \cdot \widehat{K}(x, y).$$

A continuación se presenta un esbozo de la prueba [58].

Demostración. Se parte de la definición de la DFT-2D

$$(F * K)[x, y] = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} K[u, v] \cdot F[(x - u) \bmod N, (y - v) \bmod M].$$

Al aplicar la DFT se sigue que

$$(\widehat{F * K})[k, l] = \sum_{n=0}^{M-1} \sum_{m=0}^{N-1} (F * K)[n, m] \cdot e^{-i2\pi(\frac{ml}{M} + \frac{nk}{N})}.$$

Se reemplaza la convolución en la expresión de la DFT

$$\begin{aligned} (\widehat{F * K})[m, n] &= \sum_{n=0}^{M-1} \sum_{m=0}^{N-1} \left(\sum_{u=0}^{M-1} \sum_{v=0}^{N-1} K[u, v] \cdot F[(x - u) \bmod N, (y - v) \bmod M] \right) e^{-i2\pi(\frac{ml}{M} + \frac{nk}{N})} \\ &= \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} K[u, v] \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} F[(x - u) \bmod M, (y - v) \bmod N] e^{-i2\pi(\frac{mx}{M} + \frac{ny}{N})}. \end{aligned}$$

Utilizando un cambio de variable y la periodicidad de la DFT se tiene que

$$(\widehat{F * K})[k, l] = \widehat{F}[k, l] \cdot \widehat{K}[k, l].$$

Concluyendo lo que se quería. □

No obstante, el objetivo es que partiendo de este teorema, se recupere la convolución lineal tal como se define en (11.4.2). Para ello, es necesario utilizar las condiciones descritas en la Proposición 11.4.5 previamente sobre la imagen y el núcleo. De esta manera, la convolución del teorema corresponderá con la lineal.

Este detalle no se menciona en el artículo [10], donde se describe una única convolución sin prestar demasiada atención a las dimensiones de las señales con las que se opera. En dicho artículo se sugiere reutilizar las DFTs de los elementos involucrados; sin embargo, si se quisiera ejecutar la convolución lineal podría ser necesario ser más precisos en determinados casos, ya

que las dimensiones de estas señales para aplicar el Teorema pueden variar dependiendo de con quién se convolucionan.

Esto es crucial para obtener un resultado adecuado, ya que utilizar la operación de convolución circular en el entrenamiento de CNN introduciría una artificialidad donde los bordes de la imagen se tratarían como si estuvieran directamente conectados con el lado opuesto, lo cual no es deseable en la mayoría de los casos.

11.5.1. Aplicaciones

Una de las aplicaciones que tiene este teorema es facilitar una comprensión más profunda de diversos procedimientos utilizados en VC, especialmente en operaciones como el filtrado de imágenes. Como se observa en el ejemplo de la Figura 11.4, se ha suavizado una imagen alterando su representación en el dominio de la frecuencia y luego aplicando la IDFT. Adicionalmente, también se ha mostrado que una imagen puede suavizarse mediante la convolución con un filtro Gaussiano o de paso bajo como ocurre en la Figura 11.8, lo que lleva a cuestionar cómo se relacionan estos dos procedimientos.

La conexión entre ambos reside en el Teorema de Convolución: convolucionar una imagen con un núcleo gaussiano equivale a multiplicar en el dominio de la frecuencia el filtro transformado (que vuelve a ser un filtro gaussiano, demostrado en la Observación 4.7.5) con la imagen también transformada. De este modo, el efecto de suavizado que se logra artificialmente ajustando el radio, debido a la forma del filtro gaussiano (véase Figura 11.10), se reproduce igualmente.

Esto se debe a que, al multiplicar la transformada de Fourier de la imagen punto por punto con el filtro gaussiano, se enfatiza la región central de alta amplitud de la función gaussiana. Las áreas fuera de este núcleo central se multiplican por valores mucho menores, lo que resulta en una contribución insignificante al resultado final, simulando el efecto producido en la Figura 11.4, donde el radio que se establecía en dicho ejemplo estaría constituido ahora por los valores de σ y de tamaño elegidos. Se podría decir que la convolución opera en una capa más abstracta de procesamiento y que este teorema proporciona un mayor entendimiento de esta operación.

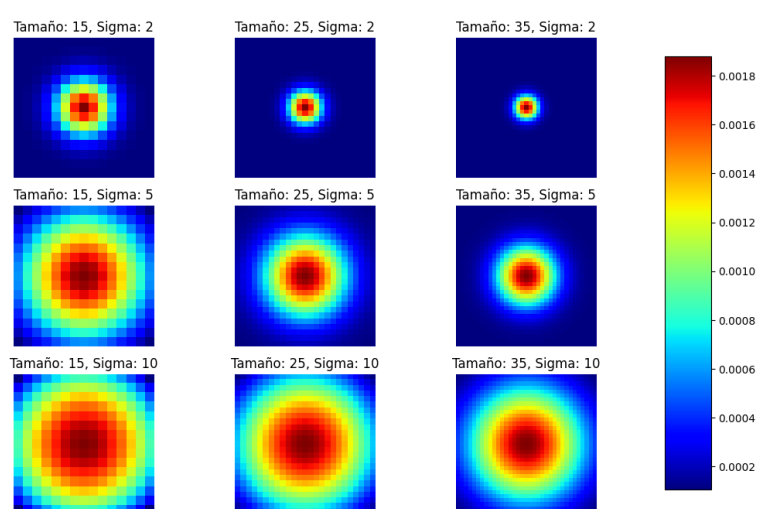


Figura 11.10.: Visualización del filtro gaussiano utilizando diferentes valores de σ y T .

Capítulo 12.

Transformada Rápida de Fourier

En el capítulo anterior se introdujo la DFT y la IDFT, destacando las ventajas de representar la señal en el dominio de la frecuencia para facilitar su análisis y procesamiento. También se explicó cómo estos conceptos se relacionan con el Teorema de la Convolution. Sin embargo, la aplicación de estas transformadas puede ser computacionalmente costosa, lo que limita su utilidad en problemas prácticos.

En este capítulo se presenta la FFT [6], un algoritmo eficiente para el cálculo de la DFT y la IDFT. Este algoritmo se encuentra prácticamente en cualquier aplicación que involucre el análisis y procesamiento de señales. Su impacto en la era moderna es incalculable, prueba de ello es su inclusión por la revista IEEE de Computación en Ciencia e Ingeniería [59] como uno de los diez algoritmos más influyentes en el desarrollo y la práctica de la ciencia y la ingeniería en el siglo XX.

El capítulo inicia con una introducción histórica del algoritmo, proporcionando un contexto sobre su evolución y desarrollo a lo largo del tiempo. A continuación, se procederá a explorar el orden de complejidad del método directo y posteriormente el fundamento teórico de la FFT, concretamente para el caso unidimensional (FFT-1D), ya que como se indicó anteriormente el cálculo de la DFT-2D se puede expresar con sucesivas aplicaciones de la DFT-1D. Este análisis teórico conducirá hacia un examen detallado de las mejoras en la eficiencia que la FFT ofrece en comparación con el método de la DFT directo. Finalmente, se presentará un algoritmo recursivo diseñado para un caso particular, demostrando cómo la FFT optimiza los cálculos en situaciones prácticas.

12.1. Historia

La Segunda Guerra Mundial finalizó en 1945 tras el lanzamiento de las primeras bombas atómicas usadas directamente sobre una población civil en Japón por parte de Estados Unidos. Este acontecimiento desencadenó en una imperiosa necesidad por parte de las naciones de controlar la proliferación nuclear ante la evidencia devastadora de su potencial destructivo.

Como resultado, Estados Unidos celebró conversaciones con los soviéticos y otras naciones con capacidad nuclear con el objetivo final de detener la carrera de armamento nuclear.

Para hacerlo se trató de llegar a determinados acuerdos que prohibían la experimentación con energía nuclear. Sin embargo, debido a la inherente desconfianza entre países, era necesario que cada nación tuviera la tecnología para identificar las pruebas nucleares realizadas por otros.

Mientras que los hidrófonos permitían la detección bajo el mar, y los residuos atómicos podían ser identificados en la atmósfera mediante pruebas terrestres, localizar pruebas subterráneas representaba un desafío significativo.

Una posibilidad consistía en analizar series temporales sismológicas obtenidas de sismómetros situados en distintos lugares estratégicos para estudiar estas señales y discernir si se trataba de un experimento nuclear. Debido a que una señal compleja comprendía múltiples ondas sinusoidales de diversas frecuencias, la aplicación de la DFT era fundamental para facilitar su análisis descomponiéndola en otras señales más sencillas. Sin embargo, la alta demanda computacional hacía inviable su aplicación con la tecnología de entonces, incluso con la de ahora.

No fue hasta 1960, cuando James Cooley y John Tukey publicaron el algoritmo de FFT [6] que reducía considerablemente el orden de complejidad de la DFT, pudiendo reducir el tiempo de cálculo que se requería de más de tres años a 35 minutos.

Trágicamente, era demasiado tarde para firmar una prohibición de pruebas completa, y las pruebas nucleares fueron forzadas bajo tierra, donde aumentaron a una notable tasa de alrededor de una vez por semana.

Desde entonces, la FFT ha encontrado nuevos usos en casi todas las aplicaciones de comunicaciones y señalización de datos. Sin embargo, su impacto podría haber sido mucho más significativo, contribuyendo a la detección de la carrera armamentista nuclear.

Lo verdaderamente notable es que la primera aparición de la FFT data de 1807 de la mano del matemático Gauss. Sus intereses estaban en ciertos cálculos astronómicos (un área recurrente de aplicación de la FFT) relacionados con la interpolación de órbitas asteroidales a partir de un conjunto finito de observaciones equidistantes. Seguramente, la perspectiva de un cálculo manual laborioso y extenso era una buena motivación para el desarrollo de un algoritmo rápido. No obstante, su trabajo no fue publicado y apareció únicamente en sus obras recopiladas como un manuscrito inédito [60].

Algunos trabajos previos en los que Cooley y Tukey se basaron fueron el desarrollo de un método eficiente para el cálculo de las interacciones de un experimento factorial 2^m introducido por Yates [61], la generalización a 3^m que fue proporcionada por Box et al. [62], y la de Good [63] que generalizó estos métodos y proporcionó algoritmos elegantes, para los cuales una clase de aplicaciones era el cálculo de series de Fourier.

Otro precursor importante es el trabajo de Danielson y Lanczos [64], realizado en el servicio de la cristalografía de rayos X, usuario frecuente de la tecnología FFT.

A lo largo de la historia, estos y numerosos otros nombres han estado vinculados al algoritmo FFT culminado con Cooley y Tukey. “Esto puede servir como motivación para explorar no solo enfoques novedosos, sino también para visitar de vez en cuando viejos documentos y descubrir la diversidad de trucos e ideas ingeniosas que se empleaban en una época en la que el cálculo era una tarea laboriosa. Quizás entre las ideas descartadas antes de la era de las computadoras electrónicas, podamos encontrar valiosas semillas para nuevos algoritmos.” [65].

12.2. Método Directo

Dada una N -tupla x , el enfoque directo para el cálculo de la DFT de x implica utilizar el método de fuerza bruta, mediante el cual, atendiendo a la Definición 11.3.1 debemos realizar una sumatoria de N sumandos para cada elemento de x . De este modo, calcular la DFT de x implica para cada $j \in \{0, \dots, N-1\}$ realizar N operaciones lo que resulta en un total de N^2 operaciones. Y, por tanto, el método directo tiene un orden de complejidad cuadrático $O(N^2)$. Razonando con la propiedad de separabilidad, o atendiendo directamente a la expresión de la DFT-2D, se deduce que para una matriz A de tamaño $N \times M$ el orden de complejidad del cálculo de la DFT-2D es $O((NM)^2)$.

Ejemplo 12.2.1. Si se dispone de una imagen de 2048×2048 (tamaño habitual), el método directo del cálculo de la DFT-2D implicaría realizar del orden de 17 billones (europeos) de operaciones excluyendo las exponenciales que podrían calcularse una única vez y almacenarse.

Observación 12.2.2. Una operación consiste en una multiplicación y una suma de dos números complejos. Denotaremos como T al número total de operaciones realizadas.

12.3. Método FFT

Sea $W_N = e^{\frac{-2\pi i}{N}}$ la N -ésima raíz de la unidad. La 1-DFT de $f = (f[0], f[1], \dots, f[N-1])$ puede reescribirse como $\hat{f} = (\hat{f}[0], \hat{f}[1], \dots, \hat{f}[N-1])$ donde:

$$\hat{f}[j] = \sum_{k=0}^{N-1} f[k] W_N^{jk}, \quad j \in \{0, 1, \dots, N-1\}. \quad (12.1)$$

Teorema 12.3.1. Sea N compuesto, con factorización $N = r_1 \cdot r_2$, donde r_1, r_2 son divisores naturales no triviales de N . La DFT de una N -tupla f puede ser calculada en $T = O(N(r_1 + r_2))$.

Demostración. Se reescriben los índices j, k como sigue:

$$\begin{aligned} j &= j_1 r_1 + j_0 \quad j_0 \in \{0, 1, \dots, r_1 - 1\}, j_1 \in \{0, 1, \dots, r_2 - 1\}, \\ k &= k_1 r_1 + k_0 \quad k_0 \in \{0, 1, \dots, r_2 - 1\}, k_1 \in \{0, 1, \dots, r_1 - 1\}. \end{aligned}$$

De esta manera, la DFT se puede expresar en función de j_0, j_1, k_0, k_1 :

$$\begin{aligned} \hat{f}[(j_1, j_0)] &= \sum_{k_0=0}^{r_2-1} \sum_{k_1=0}^{r_1-1} f(k_1, k_0) W_N^{(j_1 r_1 + j_0)(k_1 r_2 + k_0)} \\ &= \sum_{k_0=0}^{r_2-1} \sum_{k_1=0}^{r_1-1} f(k_1, k_0) W_N^{(j_1 r_1 + j_0) k_1 r_2} W_N^{(j_1 r_1 + j_0) k_0} \\ &= \sum_{k_0=0}^{r_2-1} \sum_{k_1=0}^{r_1-1} f(k_1, k_0) W_N^{j_0 k_1 r_2} W_N^{(j_1 r_1 + j_0)(k_0)}. \end{aligned} \quad (12.2)$$

CAPÍTULO 12. TRANSFORMADA RÁPIDA DE FOURIER

La última igualdad se justifica con:

$$W_N^{(j_1 r_1 + j_0) k_1 r_2} = W_N^{j_1 k_1 r_1 r_2} W_N^{j_0 k_1 r_2} = W_N^{j_1 k_1 N} W_N^{j_0 k_1 r_2} = W_N^{j_0 k_1 r_2}, \quad (12.3)$$

donde se ha usado que W_N es raíz enésima de la unidad. Es relevante destacar que $f(k_1, k_0) W_N^{j_0 k_1 r_2}$ depende únicamente de j_0 y no de k_0 . Esto permite definir la N -tupla f_1 como

$$f_1(j_0, k_0) = \sum_{k_1=0}^{r_1-1} f(k_1, k_0) W_N^{j_0 k_1 r_2}. \quad (12.4)$$

Se expresa (12.3) en función de f_1 :

$$\hat{f}[(j_1, j_0)] = \sum_{k_0=0}^{r_2-1} f_1(j_0, k_0) W_N^{(j_1 r_1 + j_0)(k_0)}. \quad (12.5)$$

Reescribiendo los índices, se han desacoplado las dos sumatorias. Esta es la clave del algoritmo de la FFT.

Como f_1 es una N -tupla y cada elemento se calcula usando r_1 operaciones, se necesitan por un lado $N r_1$ operaciones para el cálculo de f_1 . Una vez obtenido f_1 , $\hat{f}$ es calculado en r_2 operaciones por cada elemento, lo que nos deja un total de $N r_2$ operaciones. En total el algoritmo regido por las ecuaciones (12.3), (12.3) requiere un total de operaciones de $O(N(r_1 + r_2))$. $\square$

El algoritmo de la FFT planteado por Cooley-Tukey es por tanto un algoritmo de divide y vencerás que descompone recursivamente una DFT de cualquier tamaño compuesto $n = n_1 n_2$ en múltiples DFTs de menor tamaño n_1 y n_2 , aprovechando las propiedades de las raíces complejas de la unidad.

Teorema 12.3.2. *Sea N compuesto, con factorización $N = \gamma_1 \cdot \gamma_2 \cdots \gamma_m$, donde $\gamma_1, \gamma_2, \dots, \gamma_m$ son divisores naturales no triviales de N . La DFT de una N -tupla f puede ser calculada en $T = O(N(\gamma_1 + \gamma_2 + \cdots + \gamma_m))$.*

Demostración. La demostración consiste en un razonamiento inductivo sobre m , el número de factores de N .

- La etapa base $m = 2$ quedaría probada con el Teorema 12.3.1.
- Suponemos que se verifica para $m \in \mathbb{N}$, y veamos que se verifica para $m + 1$.

Sea $N = \underbrace{\gamma_1 \cdots \gamma_m}_{s_1} \cdot \underbrace{\gamma_{m+1}}_{s_2}$. Se reescriben los índices j, k como sigue:

$$j = j_1 r_1 + j_0 \quad j_0 \in \{0, 1, \dots, s_1 - 1\}, j_1 \in \{0, 1, \dots, s_2 - 1\}.$$

$$k = k_1 r_1 + k_0 \quad k_0 \in \{0, 1, \dots, s_2 - 1\}, k_1 \in \{0, 1, \dots, s_1 - 1\}.$$

Usando el mismo razonamiento que en (12.3), se concluye que

$$\widehat{f}[(j_1, j_0)] = \sum_{k_0=0}^{s_2-1} f_1(j_0, k_0) W_N^{(j_1 s_1 + j_0)(k_0)}.$$

Nótese que una vez calculado f_1 , el número de operaciones necesario para calcular $\widehat{f}$ serán s_2 operaciones por cada elemento $j_1 \in \{0, 1, \dots, s_2 - 1\}$ y $j_0 \in \{0, 1, \dots, s_1 - 1\}$, es decir en total serán $T = O(N\gamma_{m+1})$.

Al ser W_N una raíz N -ésima de la unidad y $s_1 \cdot s_2 = N$, se tiene que $(W_N)^{s_2}$ es una raíz s_1 -ésima de la unidad. Además, en f_1 el índice k_0 se encuentra fijo. Por lo que atendiendo a la expresión de f_1 y a estas consideraciones, se puede aplicar la hipótesis de inducción.

$$f_1(j_0, k_0) = \sum_{k_1=0}^{s_1-1} f(k_1, k_0) W_N^{j_0 k_1 s_2}.$$

Concluyendo entonces que para calcular f_1 hace falta $T = O(N(\gamma_1 + \dots + \gamma_m))$ operaciones. Finalmente, el número de operaciones necesarias será $T = O(N(\gamma_1 + \dots + \gamma_m + \gamma_{m+1}))$, como se quería probar. $\square$

Observación 12.3.3. Si se quiere minimizar el orden de complejidad T , se deben usar tantos factores de N como sea posible. Es decir, si $N = r_1 \cdot r_2 \cdots r_m$ y $j \in \{1, \dots, m\}$, se tiene que $r_j = p_1 p_2$ con $p_1, p_2 > 1$, y entonces $p_1 + p_2 < r_j$, salvo que $p_1 = p_2 = 2$. Por consiguiente, $O(N(r_1 + r_2 + \dots + r_j + \dots + r_m))$ tiene mayor orden de complejidad que $O(N(r_1 + r_2 + \dots + p_1 + p_2 + \dots + r_m))$. La igualdad se da para $p_1 = p_2 = 2$, de donde se deduce que el factor 2 puede combinarse de manera indistinta sin pérdida alguna.

Deducimos que los números con un mayor número de divisores proporcionarán una ganancia en eficiencia mayor. Esto motiva la siguiente definición:

Definición 12.3.4. Un número antiprimo o altamente compuesto es un entero positivo con más divisores que cualquier entero positivo más pequeño

Ejemplo 12.3.5. Ejemplos de números antiprimos son el 1, 2, 4, 6, 12, 24, 36, 48, y 60.

Proposición 12.3.6. El algoritmo FFT proporciona ganancias significativas cuando el tamaño de entrada N es un número altamente compuesto.

Demostración. El orden de complejidad de la FFT con una entrada de tamaño N con $N = p_1^{a_1} p_2^{a_2} \cdots p_m^{a_m}$ es $T = O(N(a_1 \cdot p_1 + a_2 \cdot p_2 + \dots + a_m \cdot p_m))$.

Luego,

$$\frac{T}{N} = a_1 \cdot p_1 + a_2 \cdot p_2 + \dots + a_m \cdot p_m.$$

Aplicando logaritmo,

$$\log_2 N = \log_2(p_1) \cdot a_1 + \log_2(p_2) \cdot a_2 + \dots + \log_2(p_m) \cdot a_m,$$

y así

$$\frac{T}{N \log_2(N)} = \frac{a_1 \cdot p_1 + a_2 \cdot p_2 + \dots + a_m \cdot p_m}{\log_2(p_1) \cdot a_1 + \log_2(p_2) \cdot a_2 + \dots + \log_2(p_m) \cdot a_m}.$$

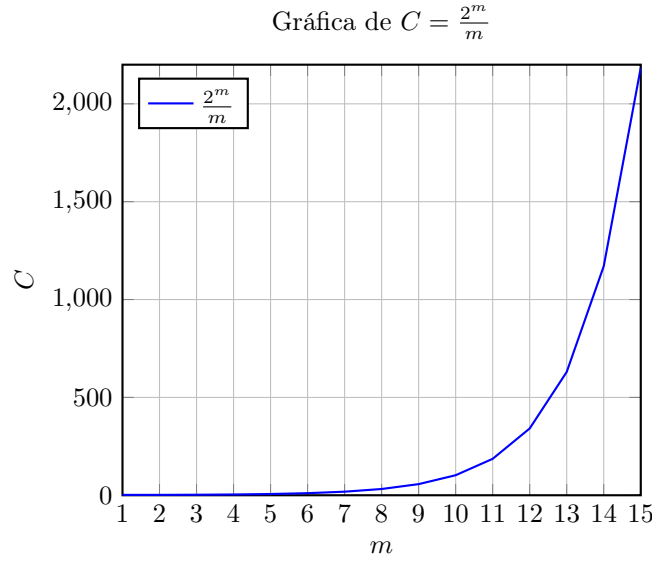


Figura 12.1.: Ganancia computacional de la FFT sobre una implementación directa de la DFT unidimensional. El número de muestras es $M_p = 2^p$. La ventaja computacional aumenta rápidamente con valores crecientes de p .

Cuando el número N es altamente compuesto, el término de la derecha se acota por una constante C , y por tanto $\frac{T}{N \log(N)} \leq C$, lo que implica que el orden de complejidad es de $O(N \log(N))$. $\square$

Proposición 12.3.7. Si $r_1 = r_2 = \dots = r_m = r$, entonces $m = \log_r N$ y el número total de operaciones para el cálculo de la FFT con tamaño de entrada $N = r^m$ es $T = O(r N \log_r N) = O(N \log_r N)$.

Demostración. Se deduce directamente del Teorema 12.3.2. $\square$

Para $N = r^m$ la ventaja computacional de utilizar la FFT en comparación con la aplicación directa de la DFT-1D se define como:

$$C(N) = \frac{N^2}{N \log_r N} = \frac{N}{\log_r N}. \quad (12.6)$$

De manera que si $N = 2^m$ ($r = 2$), se tiene que (12.3) se puede expresar en términos de m

$$C(2^m) = \frac{2^m}{m}.$$

La gráfica 12.1 de la función C muestra que la ventaja computacional aumenta rápidamente como función de m . Por ejemplo, cuando $m = 15$, la FFT tiene casi una ganancia de 2200 veces sobre la implementación de fuerza bruta de la DFT. Así, podríamos esperar que la FFT se calcule casi 2200 veces más rápido que la DFT en la misma máquina.

De hecho, el caso $r = 2$ junto con el de $r = 4$ es de especial interés a nivel computacional,

ya que el algoritmo ofrece ciertas ventajas añadidas tanto en el direccionamiento como en las operaciones realizadas, como consecuencia de la naturaleza binaria inherente a los ordenadores.

Se detalla a continuación el caso $r = 2$. La lógica seguida durante el proceso es similar a la ya aplicada anteriormente.

Sea $N = 2^m$, se expresan los índices j, k como

$$j = j_{m-1} \cdot 2^{m-1} + \dots + j_1 \cdot 2 + j_0, \quad j_i \in \{0, 1\} \quad \forall i \in \{0, \dots, m-1\},$$

$$k = k_{m-1} \cdot 2^{m-1} + \dots + k_1 \cdot 2 + k_0, \quad k_i \in \{0, 1\} \quad \forall i \in \{0, \dots, m-1\}.$$

La clave está en que esta expresión de j, k coincide con la descomposición binaria de cada índice. De tal modo que j_i, k_i se refieren al bit que ocupa la posición i de la representación binaria de j y k respectivamente, y por tanto solo pueden tomar el valor 0, 1. Escribiremos entonces (12.3) en función de sus índices. Luego las sumatorias se reducen a iterar sobre los valores 0 y 1,

$$\widehat{f}[(j_1, j_0)] = \sum_{k_0=0}^1 \sum_{k_1=0}^1 \dots \sum_{k_{m-1}=0}^1 f(k_{m-1}, \dots, k_0) W_N^{j k_{m-1} 2^{m-1} + \dots + j k_0}.$$

Siguiendo un razonamiento similar a (12.3), como consecuencia de $W_N^N = 1$, se tiene:

$$W_N^{j k_{m-1} 2^{m-1}} = W_N^{(j_{m-1} \cdot 2^{m-1} + \dots + j_1 \cdot 2 + j_0) k_{m-1} 2^{m-1}} = W_N^{j_0 k_{m-1} 2^{m-1}}.$$

Esto permite desacoplar la suma interior f_1 y calcularla directamente como

$$f_1(j_0, k_{m-2}, \dots, k_0) = \sum_{k_{m-1}=0}^1 f(k_{m-1}, \dots, k_0) W_N^{j_0 k_{m-1} 2^{m-1}},$$

ya que sólo depende de j_0 .

Se procede de igual forma con el resto de índices desacoplando las distintas sumatorias. De esta manera, en general para $l \in \{1, \dots, m\}$ se tendría

$$f_l(j_0, \dots, j_{l-1}, k_{m-l-1}, \dots, k_0) = \sum_{k_{m-l}=0}^1 f_{l-1}(j_0, \dots, j_{l-2}, k_{m-l}, \dots, k_0) W_N^{(j_{l-1} 2^{l-1} + \dots + j_0) k_{m-l} 2^{m-l}},$$

donde se ha usado

$$W_N^{j k_{m-l} 2^{m-l}} = W_N^{(j_{m-1} \cdot 2^{m-1} + \dots + j_1 \cdot 2 + j_0) k_{m-l} 2^{m-l}} = W_N^{j_0 k_{m-l} 2^{m-l}}.$$

Desarrollando la sumatoria y usando que $e^{i\pi} = -1$, $e^{i\frac{\pi}{2}} = i$,

$$\begin{aligned} f_l(j_0, \dots, j_{l-1}, k_{m-l-1}, \dots, k_0) \\ = f_{l-1}(j_0, \dots, j_{l-2}, 0, k_{m-l-1}, \dots, k_0) \end{aligned} \quad (12.7)$$

$$+ (-1)^{j_{l-1}} i^{j_{l-2}} f_{l-1}(j_0, \dots, j_{l-2}, 1, k_{m-l-1}, \dots, k_0) W_N^{(j_{l-3} 2^{l-3} + \dots + j_0) 2^{m-l}} \quad j_{l-1} \in \{0, 1\}.$$

De acuerdo con la convención de indexación, esto se almacena en una ubicación cuyo índice es

$$j_0 2^{m-1} + \dots + j_{l-1} 2^{m-l} + k_{m-l-1} 2^{m-l-1} + \dots + k_0.$$

Se puede observar en (12.3) que solo están involucradas en el cálculo las dos ubicaciones de almacenamiento con índices que tienen 0 y 1 en la posición del bit 2^{m-1} .

Además la computación paralela está permitida, ya que la operación descrita por (12.3) se puede llevar a cabo con todos los valores de $j_0, \dots, j_{l-2}$ y $k_0, \dots, k_{m-l-1}$ simultáneamente. En algunas aplicaciones, es conveniente usar (12.3) para expresar A_l en términos de A_{l-2} , lo que equivale a un algoritmo con $r = 4$.

El último array f_m calculado proporciona el resultado deseado de la siguiente manera.

$$\hat{f}(j_{m-i}, \dots, j_0) = f_m(j_0, \dots, j_{m-1}).$$

De tal manera que el orden del índice en $\hat{f}$ debe tener sus bits colocados en orden inverso para poder acceder al índice en la matriz f .

Ejemplo 12.3.8. En el Ejemplo 12.2.1, se muestra que para calcular la FFT 2-D de una imagen de 2048×2048 a través del método directo se requieren alrededor de 17 billones (europeos) de operaciones. Usando la propiedad de separabilidad, la DFT-2D se puede calcular a partir de sucesivas DFT 1-D, concretamente $(2048 + 2048)$, y usando el algoritmo de la FFT para cada una de estas, junto con que $2048 = 2^{11}$, se tiene que el número de operaciones es del orden de 92 millones, lo cual representa una reducción significativa respecto a los cálculos mencionados anteriormente.

12.4. Algoritmos FFT

En la sección anterior se ha detallado el algoritmo FFT original y el cálculo de su eficiencia, aspecto que es de particular interés para estudiar el método alternativo de convolución que se propone en este trabajo.

Sin embargo, a raíz de este algoritmo, surgieron numerosas variaciones que han dado lugar a un gran número de implementaciones, constituyendo toda una familia de algoritmos de FFT cuyo principio general es utilizar una estrategia de divide y vencerás para factorizar la matriz W en submatrices más pequeñas, que es precisamente la clave que se utiliza en el desarrollo anterior (lo que permitía desacoplar las sumatorias). De manera que a pesar de que cada algoritmo siga una estrategia diferente para agrupar las sumatorias, la idea que subyace a todos ellos es la presentada con anterioridad. Todos los algoritmos FFT conocidos requieren $O(n \log n)$ operaciones, aunque no hay ninguna prueba conocida de que una complejidad menor sea imposible.

12.4.1. Radix-2

Uno de los algoritmos de la FFT más conocidos es el Radix-2 de decimación en tiempo (DIT), también conocido simplemente como **Radix-2**. Esta es una implementación recursiva del algoritmo FFT diseñada específicamente para cuando $N = 2^\gamma$, y aunque es la forma más simple y común del algoritmo de Cooley-Tukey, sigue siendo ampliamente utilizada.

Radix-2 divide una DFT de tamaño N en dos DFTs entrelazadas de tamaño $\frac{N}{2}$ en cada etapa recursiva. Esto se logra seleccionando alternativamente los índices pares e impares. Este procedimiento se visualiza en la Figura 12.3. En efecto, si partimos de un vector $f = [f[0], f[1], \dots, f[N-1]]$, y expresamos cada índice par como $k = 2m$ y cada índice impar como $k = 2m + 1$, entonces se pueden formar dos subvectores para la recursividad.

$$\hat{f}[j] = \sum_{k=0}^{N-1} f[k] W_N^{jk} = \underbrace{\sum_{m=0}^{\frac{N}{2}-1} f[2m] W_N^{2mj}}_{\text{parte par}} + \underbrace{\sum_{m=0}^{\frac{N}{2}-1} f[2m+1] W_N^{j(2m+1)}}_{\text{parte impar}}, \quad j \in \{0, 1, \dots, N-1\}.$$

Usando la expresión de $W_N = e^{-\frac{2\pi i}{N}}$, se tiene para $j \in \{0, 1, \dots, N-1\}$

$$\hat{f}[j] = \underbrace{\sum_{m=0}^{\frac{N}{2}-1} f[2m] e^{-\frac{2\pi i}{N}mj}}_{\text{DFT de la parte par de } f} + e^{-\frac{2\pi i}{N}j} \underbrace{\sum_{m=0}^{\frac{N}{2}-1} f[2m+1] e^{-\frac{2\pi i}{N}mj}}_{\text{DFT de la parte impar de } f} = E_j + e^{-\frac{2\pi i}{N}j} O_j.$$

Nótese que las igualdades son válidas para $k \in \{0, \dots, N-1\}$, pero la clave es que E_j y O_j son calculadas para $j \in \{0, 1, \dots, \frac{N}{2}-1\}$.

Gracias a la periodicidad de la exponencial compleja, $\hat{f}[j + \frac{N}{2}]$ también se obtiene de E_j y O_j para $j \in \{0, 1, \dots, \frac{N}{2}-1\}$:

$$\begin{aligned} \hat{f}\left[j + \frac{N}{2}\right] &= \sum_{m=0}^{\frac{N}{2}-1} f[2m] e^{-\frac{2\pi i}{N}m(j+\frac{N}{2})} + e^{-\frac{2\pi i}{N}(j+\frac{N}{2})} \sum_{m=0}^{\frac{N}{2}-1} f[2m+1] e^{-\frac{2\pi i}{N}m(j+\frac{N}{2})} \\ &= \sum_{m=0}^{\frac{N}{2}-1} f[2m] e^{-\frac{2\pi i}{N}mj} + e^{-\frac{2\pi i}{N}j} e^{-\pi i} \sum_{m=0}^{\frac{N}{2}-1} f[2m+1] e^{-\frac{2\pi i}{N}mj} e^{-2\pi im} \\ &= \sum_{m=0}^{\frac{N}{2}-1} f[2m] e^{-\frac{2\pi i}{N}mj} - e^{-\frac{2\pi i}{N}j} \sum_{m=0}^{\frac{N}{2}-1} f[2m+1] e^{-\frac{2\pi i}{N}mj} = E_j - e^{-\frac{2\pi i}{N}j} O_j. \end{aligned}$$

Por lo que se tendría para $j \in \{0, 1, \dots, \frac{N}{2}-1\}$

$$\begin{aligned} \hat{f}[j] &= E_j + e^{-\frac{2\pi i}{N}j} O_j, \\ \hat{f}\left[j + \frac{N}{2}\right] &= E_j - e^{-\frac{2\pi i}{N}j} O_j. \end{aligned} \tag{12.8}$$

Este resultado, que expresa la DFT de longitud N de forma recursiva en términos de dos DFTs

de tamaño $N/2$, constituye la esencia del algoritmo de la FFT radix-2 DIT. La Figura 12.2 visualiza el cálculo (12.4.1). Usando esta idea se construye el algoritmo recursivo detallado en Algoritmo 2.

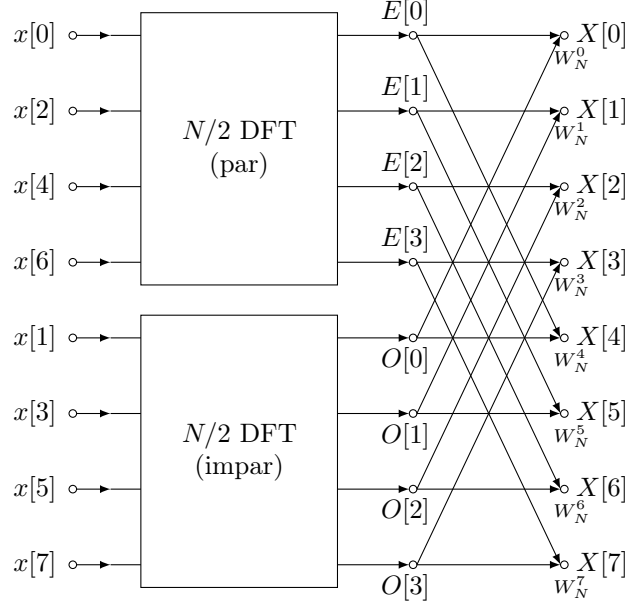


Figura 12.2.: Diagrama del algoritmo Radix-2 DIT para $N = 8$.

Seguidamente, se desarrollaron otros algoritmos, como **Radix-3** (en el que cada DFT se divide en tres DFTs de tamaño $\frac{N}{3}$), **Radix-4** (en el que cada DFT se divide en cuatro DFTs de tamaño $\frac{N}{4}$), y **Mixed Radix** que es una variante que usa los anteriores y permite manejar tamaños de muestra que no son potencias de un solo número primo, como en los casos de Radix-2, Radix-3 o Radix-4. Mixed Radix FFT es particularmente útil para secuencias cuyas longitudes son productos de diferentes factores primos.

La biblioteca cuFFT [67], conocida por su eficiente implementación de la FFT, implementa los siguientes bloques básicos: radix-2, radix-3, radix-5 y radix-7. Por lo tanto, el rendimiento de la DFT para cualquier tamaño que pueda factorizarse como

$$2^a \times 3^b \times 5^c \times 7^d$$

(donde a, b, c , y d son enteros no negativos) está optimizado en la biblioteca cuFFT, y se trata de expresar el tamaño de entrada usando esta descomposición. También existen bloques de construcción radix- m para otros primos, m , cuyo valor es menor que 128.

Existen numerosos otros algoritmos, de los que destacan el algoritmo de **Factor Primo** (PFA) [63] basado en el teorema del resto chino para una descomposición de N en coprimos, o la **FFT Hexagonal** (HFFT) [68] para datos muestreados de forma hexagonal utilizando un nuevo esquema de direccionamiento para las rejillas hexagonales, denominado direccionamiento de conjunto de arrays (ASA).

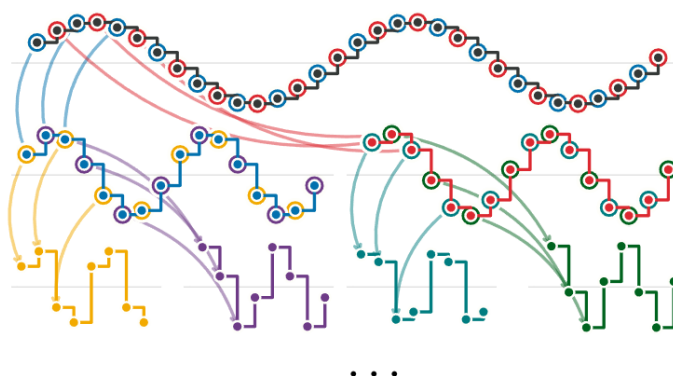


Figura 12.3.: Una señal de $N = 32$ se divide en muestras pares e impares (mitad izquierda y mitad derecha). Cada una de estas a su vez, pueden dividirse de manera similar (fila inferior). Imagen obtenida de [66].

Como se puede observar, existe una extensa variedad de algoritmos dentro de la familia de la FFT, aunque, como se ha mencionado anteriormente, muchas de las implementaciones se fundamentan en los métodos de descomposición Radix.

Por último, nótese que el algoritmo de Cooley-Tukey original de la FFT no contempla para N primo ninguna mejora. De hecho, una posible estrategia en ese caso es aplicar un zero-padding a la potencia de 2 más cercana para aproximar la DFT. Sin embargo, si lo que se quiere es calcular la DFT de una secuencia de elementos para un valor de N general (incluido para N primo), se podrían recurrir al algoritmo de **Rader** [69], que reescribe la DFT como una convolución circular, o el de **Bluestein** [70] que puede utilizarse para calcular transformadas más generales que la DFT, basándose en la transformada z unilateral. En la biblioteca cuFFT cuando la longitud no puede descomponerse como múltiplos de potencias de primos de 2 a 127, se utiliza el algoritmo de Bluestein.

Algoritmo 2 FFT (Radix-2)

procedure FFT(A)**Entrada:** Un array de valores complejos con tamaño 2^m para $m \geq 0$.**Salida:** Un array de valores complejos que es la DFT de la entrada. $N := \text{longitud}(A)$ **Si** $N = 1$ **entonces****devolver** A **si no** $W_N := e^{\frac{2\pi i}{N}}$ $W := 1$ $A_{\text{par}} := (A_0, A_2, \dots, A_{N-2})$ $A_{\text{impar}} := (A_1, A_3, \dots, A_{N-1})$ $Y_{\text{par}} := \text{FFT}(A_{\text{par}})$ $Y_{\text{impar}} := \text{FFT}(A_{\text{impar}})$ **Para** $j := 0$ **hasta** $\frac{N}{2} - 1$ **hacer** $Y[j] = Y_{\text{par}}[j] + W \cdot Y_{\text{impar}}[j]$ $Y[j + \frac{N}{2}] = Y_{\text{par}}[j] - W \cdot Y_{\text{impar}}[j]$ $W := W \cdot W_N$ **Fin Para****devolver** Y **Fin Si****Fin procedure**

Capítulo 13.

Análisis de la Eficiencia

En este capítulo se realiza un análisis de la eficiencia temporal del método de convolución propuesto y del proceso de entrenamiento de una CNN íntegramente en el dominio de frecuencia. Este tipo de análisis, a diferencia del análisis experimental, no depende de las implementaciones específicas o del hardware utilizado y ofrece información crucial para elegir entre dos algoritmos o métodos para la resolución de un determinado problema.

13.1. Método Alternativo de la Convolución

A continuación se detalla el algoritmo de convolución propuesto, el cual emplea el Teorema de Convolución. Se realizará una comparación entre este y el algoritmo de convolución tradicional descrito en el Capítulo 11.

Se usará para esta discusión el análisis del algoritmo de la FFT en el Capítulo 12.

Se supondrá que tanto la imagen como el *kernel* son cuadrados, aunque el análisis se extiende de manera natural para el caso en el que no lo son. Ambos algoritmos de convolución devuelven una matriz con las mismas dimensiones que la entrada, es decir, utilizando el modo “same” presentado en el Capítulo 11. Se describen a continuación los pasos del algoritmo:

1. Transformar la imagen $n \times n$ y el *kernel* $k \times k$ al dominio de Fourier utilizando la FFT. Para ello, el *kernel*, que generalmente es mucho más pequeño que la imagen, se ajusta al mismo tamaño que esta. En general, para evitar propiedades de envolvimiento no deseadas como se comentó en el Capítulo 11, se redimensionan ambos operandos a un tamaño $N > n + k - 1$, que suele ser potencia de dos para aprovechar las propiedades del algoritmo FFT en este caso. Para facilitar la comparativa, se supondrá que $N \approx n$, ya que al trabajar con datos de grandes dimensiones, $n - k + 1 \approx n$.
2. Realizar el producto punto a punto de las transformadas.
3. Transformar el resultado de vuelta al dominio espacial utilizando la IDFT. Para ello se deberá reajustar la salida al tamaño deseado.

Las tres DFT que se ven involucradas en el algoritmo requieren:

$$O(2n^2 \log n^2) = O(2n^2 \log n) = 6Cn^2 \log n \text{ operaciones,}$$

donde C representa las constantes ocultas en la notación $O(\cdot)$, y el producto punto a punto en

el dominio de la frecuencia requiere

$$4n^2 \text{ operaciones.}$$

Nótese que el producto es entre dos números complejos, de donde procede la constante 4.

Por tanto, la comparativa entre ambos métodos se muestra en la siguiente Tabla:

Convolución Directa	Método Propuesto
$(n - k + 1)^2 k^2$	$6Cn^2 \log n + 4n^2$

Nótese los siguientes aspectos:

- La complejidad del método directo de convolución depende del tamaño del *kernel*, y puede ser lenta para *kernels* grandes o no separables.
- La complejidad del algoritmo de convolución propuesto es independiente del tamaño del filtro. Esto se puede visualizar en la Figura 13.2.

Por tanto, la convolución en el dominio de la frecuencia es típicamente más eficiente que la convolución directa cuando el tamaño del filtro supera un umbral particular. Así, para filtros relativamente pequeños, la convolución directa es más eficiente, mientras que para filtros más grandes, llega un punto en el que la convolución basada en FFT es más eficiente. La Figura 13.1 presenta una gráfica comparativa para diversos tamaños de kernel $k \times k$ y distintos tamaños de imágenes $n \times n$. Observando estas gráficas, se observa que a partir de $k = 12$ el método propuesto realiza un menor número de operaciones en comparación con el método tradicional.

13.2. Entrenamiento en el Dominio de la Frecuencia

Se describe en el Capítulo 9 la estructura de una CNN, que se compone generalmente de tres tipos de capas: capas convolucionales, capas de Pooling y capas totalmente conectadas. Sin embargo, la capa en la que el método de convolución propuesto tiene un impacto más significativo es, sin duda, en la capa convolucional; el resto se pueden simular en el dominio de la frecuencia sin un coste significativamente mayor.

Cada capa convolucional realiza tres tareas fundamentalmente. Estas se describen en la Subsección 10.2.1. A continuación, se realiza una comparativa de eficiencia de realizar dichas tareas en el dominio de la frecuencia y realizarla en el dominio espacial.

Se fija la siguiente notación para el análisis: para una capa dada, se dispone un conjunto de mapas de características de entrada x_f indexados por f , cada uno de los cuales es una imagen 2-D de dimensiones $n \times n$. La salida es un conjunto de mapas de características $y_{f'}$ indexados por f' , que también son imágenes 2-D cuya dimensión depende del kernel de convolución y su paso. Los parámetros entrenables de la capa consisten en un conjunto de pesos $w_{f'f}$, cada uno de los cuales es un pequeño kernel de dimensiones $k \times k$. Por último, $n' = (n - k + 1)$ representa el tamaño del mapa de características de salida. De manera que las tareas descritas quedarían con esta notación como:

$$y_{f'} = \sum_f x_f * w_{f'f},$$

13.2. ENTRENAMIENTO EN EL DOMINIO DE LA FRECUENCIA

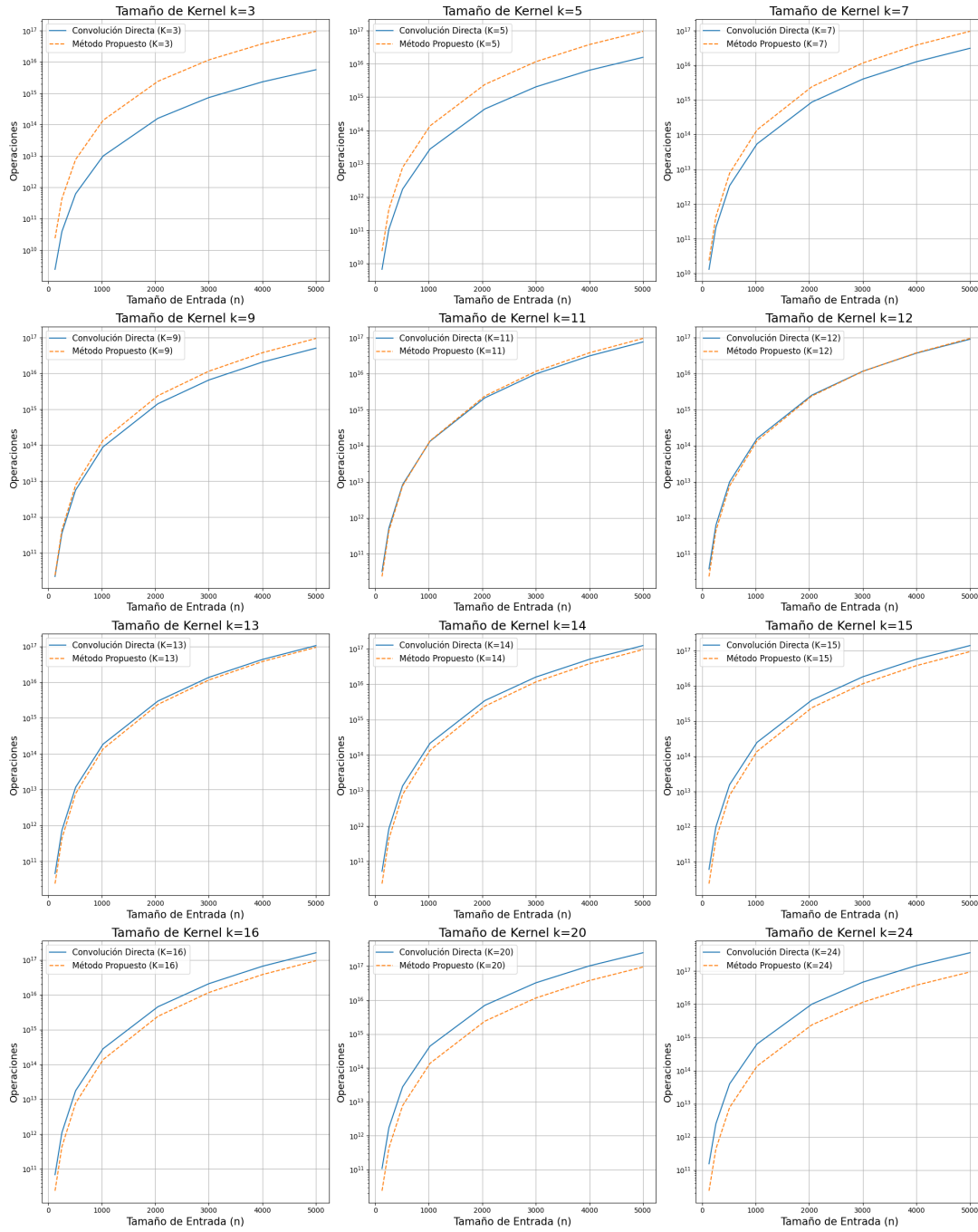


Figura 13.1.: Se muestra el número teórico de operaciones para la convolución directa y el método FFT propuesto para varios tamaños de entrada y de *kernel*.

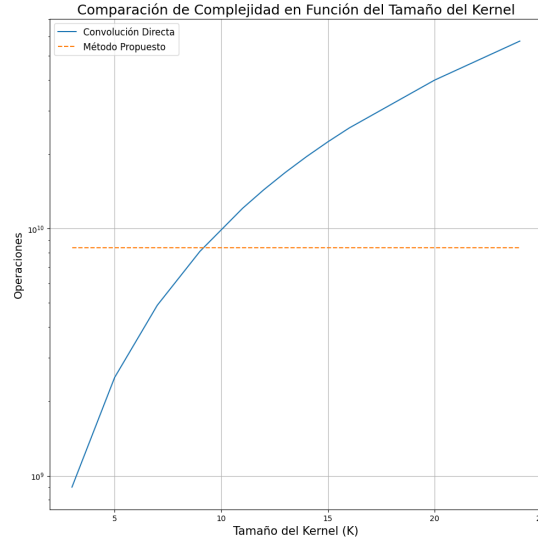


Figura 13.2.: Comparativa de la complejidad de ambos métodos en función del tamaño del núcleo.

$$\frac{\partial L}{\partial x_f} = \frac{\partial L}{\partial y_{f'}} * w_{f'f}^T,$$

$$\frac{\partial L}{\partial w_{f'f}} = \frac{\partial L}{\partial y_{f'}} * x_f,$$

Tarea	Convolución Directa	Método Propuesto
$\sum_f x_f * w_{f'f}$	$S \cdot f' \cdot f \cdot n'^2 \cdot k^2$	$2Cn^2 \log n [f \cdot S + f' \cdot f] + 4S \cdot f' \cdot f \cdot n^2$
$\frac{\partial L}{\partial y_{f'}} * w_{f'f}^T$	$S \cdot f' \cdot f \cdot n^2 \cdot k^2$	$2Cn'^2 \log n' [f \cdot S + f' \cdot f] + 4S \cdot f' \cdot f \cdot n'^2$
$\frac{\partial L}{\partial y_{f'}} * x_f$	$S \cdot f' \cdot f \cdot k^2 \cdot n'^2$	$2Cn^2 \log n [f \cdot S + f' \cdot f] + 4S \cdot f' \cdot f \cdot n^2$

donde el texto resaltado en azul corresponde al número de operaciones necesarias para realizar la transformación al dominio de Fourier de los operandos implicados en cada tarea, y en rojo, las relativas a las sumas y multiplicaciones en el dominio de Fourier.

Si se atiende a las tareas descritas, cada una de las matrices indexadas por f se convoluciona con cada una de las matrices indexadas por f' . Por lo tanto, se podría calcular la FFT de cada matriz una vez, y todas las convoluciones por pares pueden realizarse como productos en el dominio de la frecuencia. Por lo que, aunque para realizar una convolución, el uso del método basado en FFT puede ser menos eficiente que el directo, en el primer caso se pueden reutilizar las FFT calculadas en sucesivas ocasiones, lo que compensa con creces la sobrecarga.

Nótese que la complejidad del método directo de convolución proviene del producto de cinco términos, que dependen del tamaño del *kernel*, mientras que el método propuesto, independiente del tamaño del *kernel*, tiene una suma de productos con un máximo de cuatro términos. Este análisis se presenta en el artículo de referencia [10]. La diferencia radica en que dicho estudio incluye, en cada tarea, las operaciones necesarias para la ejecución de la DFT inversa para volver al dominio espacial, mientras que con la aproximación propuesta esto no es necesario en

cada etapa. Sin embargo, al final de las capas, es necesario volver al dominio espacial, ya sea aplicando la DFT inversa u otra operación que permita interpretar el resultado en términos de magnitudes reales.

Como consideración final, esta aproximación no requiere un uso intensivo de la memoria. Para una capa de convolución que toma una entrada de tamaño $n \times n$, con f características de entrada, f' características de salida y un minibatch de tamaño S , se necesita almacenar un total de $S \cdot f + S \cdot f' + f \cdot f'$ representaciones en frecuencia de tamaño $n \times n$. Se pueden usar las propiedades de simetría de las FFTs de entradas reales para almacenar solo la mitad de los datos, es decir, $n(n+1)/2$ números complejos. Suponiendo representaciones en punto flotante, la memoria necesaria en bytes es:

$$4n(n+1)(S \cdot f + f \cdot f').$$

Nótese que este es un requisito adicional de memoria relativamente pequeño en comparación con la cantidad total de memoria utilizada en redes profundas.

Capítulo 14.

Análisis Experimental

En este capítulo se ejecutan una serie de experimentos que complementan y validan las teorías desarrolladas a lo largo del presente TFG. Es importante resaltar que, dado que el código del artículo de referencia [10] no está disponible, se ha procurado seguir la misma metodología y enfoque experimental. Por tanto, se han diseñado una serie de pruebas para evaluar la capacidad del algoritmo de convolución desarrollado. Estas pruebas incluyen una primera parte de experimentos tanto de la aplicación del mismo para problemas reales como comparativos entre el algoritmo propuesto y otros algoritmos existentes para analizar sus eficiencias relativas. Posteriormente, existe una segunda parte donde se ha integrado el nuevo algoritmo en una CNN para compararla con una arquitectura clásica.

Este análisis está sujeto a la implementación específica y el entorno de operación, incluyendo el hardware y el software. Esto puede hacer que los resultados sean específicos para la configuración particular utilizada en los experimentos.

14.1. Entorno de Desarrollo

Google Colab¹ es una plataforma gratuita de Google que permite a los usuarios escribir y ejecutar código Python en el navegador, con una configuración mínima y sin necesidad de instalar software adicional. Es una herramienta especialmente útil para la investigación y el desarrollo en el campo del AA y la ciencia de datos.

Una de las características más destacadas de Google Colab es la posibilidad de utilizar unidades de procesamiento tensorial (TPU). Las TPUs son aceleradores de hardware diseñados específicamente por Google para optimizar el rendimiento de las operaciones de DL. Ofrecen una potencia de procesamiento significativamente mayor en comparación con las unidades de procesamiento gráfico (GPU) y las unidades centrales de procesamiento (CPU), lo que puede reducir drásticamente el tiempo necesario para entrenar modelos complejos de AA. A continuación se muestra el hardware usado en el análisis experimental

- RAM Total: 12.7 GB.
- RAM de la GPU: 15 GB.
- Disco: 78.2 GB.
- $6 \times$ CPU Intel(R) Xeon(R) CPU @ 2.00GHz.

¹<https://colab.research.google.com>

- GPU NVIDIA Tesla T4.

En este entorno de desarrollo, se utilizarán varias librerías de Python que son esenciales para el procesamiento y análisis de datos, así como para la implementación de modelos de AA. A continuación se describen brevemente estas librerías y las versiones utilizadas:

- **NumPy**

NumPy (Numerical Python) es una biblioteca fundamental para la computación científica en Python. Proporciona soporte para arrays y matrices multidimensionales, junto con una colección de funciones matemáticas de alto nivel para operar con ellos. NumPy está optimizado para realizar operaciones numéricas complejas de manera rápida y eficiente. Se utilizará la versión 1.25.2.

- **SciPy**

SciPy (Scientific Python) es una biblioteca que extiende las capacidades de NumPy con funciones adicionales para la optimización, integración, interpolación, álgebra lineal, estadísticas y otras tareas científicas y técnicas. Se utilizará la versión 1.11.4.

- **OpenCV**

OpenCV (Open Source Computer Vision Library) es una biblioteca de VC que proporciona herramientas para la captura, procesamiento y análisis de imágenes y videos. OpenCV es altamente optimizada y soporta múltiples plataformas. Se utilizará la versión 4.8.0.

- **Matplotlib**

Matplotlib es una biblioteca para la creación de visualizaciones estáticas, animadas e interactivas en Python. Se utilizará la versión de 3.7.1.

- **TensorFlow**

TensorFlow es una biblioteca de código abierto desarrollada por Google para el AA y el DL. Proporciona una amplia gama de herramientas y recursos para construir y entrenar modelos de redes neuronales. Se utilizará la versión 2.15.0.

- **Keras**

Keras es una biblioteca de alto nivel para el DL que se ejecuta sobre TensorFlow. Proporciona una API sencilla y fácil de usar para construir y entrenar modelos de DL. Se utilizará la versión 2.15.0.

El código desarrollado se encuentra en notebooks de Jupyter, a los que se puede acceder a través del siguiente enlace de Github².

14.2. Método Alternativo de la Convolución.

A continuación, se implementa el método alternativo de la convolución `convolution_fft` utilizando la FFT, descrito en el Capítulo 13.

En el Algoritmo 3 se presenta un esbozo de la implementación para realizar la convolución en imágenes en escala de grises. Para imágenes a color, simplemente se aplica este algoritmo a cada canal por separado. Véase el Algoritmo 4.

²<https://github.com/isa5456/TFG>

Algoritmo 3 *convolution_fft Imagen en escala de grises.***Entrada:** imagen, kernel**Salida:** resultado_convolucion

- 1: $H, W \leftarrow$ dimensiones de *imagen*
- 2: $k_1, k_2 \leftarrow$ dimensiones de *kernel*
- 3: $fftsize \leftarrow (2^{\lceil \log_2(H+k_1-1) \rceil}, 2^{\lceil \log_2(W+k_2-1) \rceil})$
- 4: $image_fft \leftarrow \text{fft2}(imagen, fftsize)$
- 5: $kernel_fft \leftarrow \text{fft2}(kernel, fftsize)$
- 6: $result_fft \leftarrow image_fft \times kernel_fft$
- 7: $result_inversa \leftarrow \text{ifft2}(result_fft)$
- 8: $result_inversa \leftarrow \text{real}(result_inversa)$
- 9: $result_convolucion \leftarrow result_inversa[k_1/2 : k_1/2 + H, k_2/2 : k_2/2 + W]$
- 10: **Devolver** $result_convolucion$

Algoritmo 4 *convolution_fft_multiple_channels Imagen a color RGB.***Entrada:** imagen, kernel**Salida:** resultado_convolucion

- 1: $resultado_convolucion = \mathbf{0}_{m \times n \times c}$
- 2: **Para** cada canal en el rango del número de canales en *imagen* **hacer**
- 3: $resultado_convolucion[:, :, canal] = \text{convolution_fft}(imagen[:, :, canal], kernel)$
- 4: **Fin Para**
- 5: **Devolver** $resultado_convolucion$

14.2.1. Experimentación con el Algoritmo Propuesto. Primera Parte

A continuación se presentan una serie de experimentos realizados para evaluar la viabilidad del nuevo método de convolución *convolution_fft* en problemas de VC. Adicionalmente, a través de estos experimentos se pone de manifiesto la utilidad de esta operación en aplicaciones más allá del DL.

14.2.1.1. Pirámides.

- **Pirámide Gaussiana.** Se plantea construir una pirámide Gaussiana [71], una técnica crucial en el procesamiento de imágenes y VC que se utiliza para representar una imagen en múltiples escalas de resolución. La pirámide Gaussiana es ampliamente utilizada en aplicaciones de procesamiento de imágenes, como la detección de características, el análisis multi-resolución y la compresión de imágenes, debido a su capacidad para preservar la información relevante de la imagen a diferentes niveles de detalle.

El mecanismo de construcción de la pirámide Gaussiana consta de dos etapas: una de suavizado (en la que se utiliza el algoritmo de convolución) y otra de reducción de dimensión.

En la parte de suavizado, se utiliza la convolución descrita con un filtro Gaussiano, lo cual permite suavizar la imagen eliminando detalles finos y ruido. Posteriormente, en la etapa de reducción de dimensión, la imagen suavizada se recorta, típicamente eliminando filas y

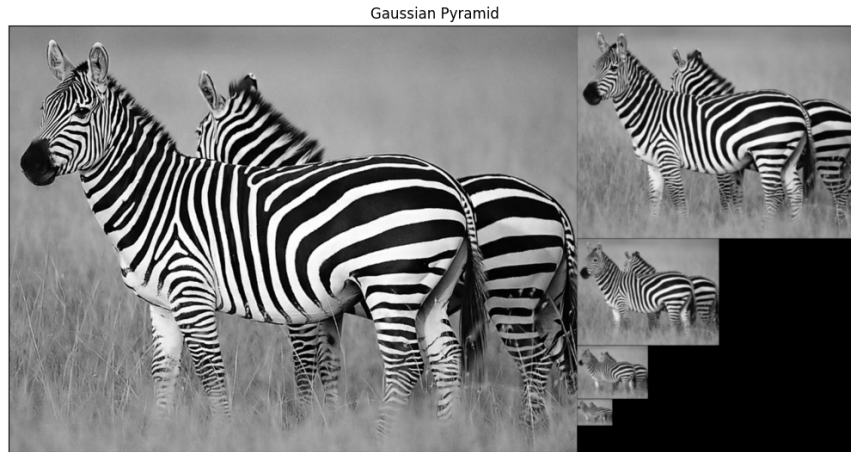


Figura 14.1.: Ejemplo de pirámide Gaussiana.

columnas pares o mediante una función de redimensionado, reduciendo así la resolución de la imagen.

Este proceso se realiza de manera iterativa hasta alcanzar el número de niveles deseado en la pirámide. La importancia del suavizado para reducir la dimensionalidad es notoria; al recortar la imagen sin suavizar, aparece ruido y distorsión en la misma, lo cual se evita mediante el suavizado. De esta manera, se obtiene una versión suavizada de la imagen a distintas escalas, permitiendo una representación jerárquica y multi-resolución de la misma. Al suavizar y reducir la imagen, se preserva la información global mientras se eliminan detalles finos. En la Figura 14.1 se muestra un ejemplo de Pirámide Gaussiana.

- **Pirámide Laplaciana.** Se pide construir una pirámide Laplaciana [72], que se calcula a partir de la Gaussiana. La pirámide Laplaciana destaca los detalles de alta frecuencia en cada nivel, lo cual es útil para aplicaciones que requieren detectar y representar características finas.

El proceso es el siguiente: Se parte de la imagen más pequeña producida por la Pirámide Gaussiana (como la gaussiana detecta los detalles gruesos, la diferencia calcula los detalles finos de esta), se expande y se calcula la diferencia con el siguiente nivel de la Gaussiana. Este proceso se repite de manera iterativa y estas diferencias van constituyendo la pirámide Laplaciana, a la que luego se le debe dar la vuelta, ya que a diferencia de la Gaussiana, esta es construida de menor a mayor tamaño. En la Figura 14.2 se muestra un ejemplo de una pirámide Laplaciana. A diferencia de la pirámide Gaussiana, en esta se almacenan los detalles finos a cada nivel, así como los bordes y las texturas de la imagen.

A partir de la pirámide Laplaciana, si esta se ha construido correctamente, se puede recuperar **exactamente** la imagen original [72]. Este proceso parte de la imagen de menor dimensión en la pirámide Laplaciana. Esta imagen se aumenta hasta que coincida con las dimensiones de la imagen Laplaciana del siguiente nivel superior. A continuación, se suman ambas imágenes. Este procedimiento se repite iterativamente, subiendo de nivel, hasta que se alcanza y suma la imagen del último nivel de la pirámide Laplaciana. Al final, se obtiene la imagen original. En

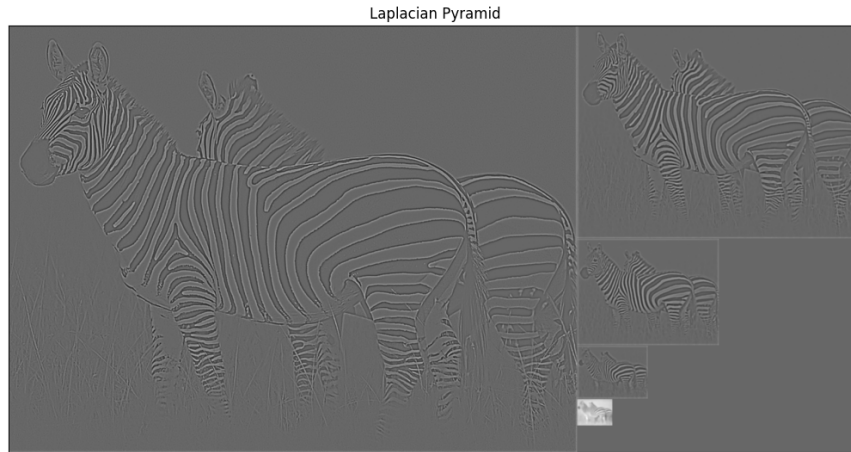


Figura 14.2.: Ejemplo de pirámide Laplaciana.

la Figura 14.3 se muestra la reconstrucción de la imagen a partir de la Pirámide Laplaciana mostrada en la Figura 14.2.

Este método no solo es una técnica eficaz para la reconstrucción de imágenes, sino que también proporciona una manera de comprobar la precisión y efectividad de nuestro método de convolución. Si la imagen original se puede reconstruir con exactitud a partir de la pirámide Laplaciana, esto indica que nuestro método de convolución ha funcionado de manera precisa y efectiva en cada etapa de la descomposición y posterior reconstrucción de la imagen. Este enfoque abre un abanico de posibilidades, donde en ocasiones resulta más ventajoso tener almacenadas, en lugar de la imagen original, las “piezas” necesarias para poder reconstruirla.

14.2.1.2. Imágenes Híbridas.

Este ejercicio se basa en el trabajo de Aude Oliva, Antonio Torralba y Philippe. G. Schyns [73]. El objetivo de este es aprender cómo el sistema visual humano extrae información sobre un objeto en función de la distancia. Para ello, se construye una imagen híbrida a partir de dos imágenes de objetos diferentes. Mezclando adecuadamente parte de las altas frecuencias de una imagen con parte de las bajas frecuencias de otra imagen, se obtiene una imagen híbrida que muestra diferentes percepciones con la distancia. El σ empleado para filtrar ambas imágenes (tanto la de alta como la de bajas frecuencias) es el aspecto clave para seleccionar el rango de frecuencias altas y bajas de cada imagen. Cuanto mayor sea el valor σ , mayor será la eliminación de altas frecuencias de la imagen.

Es también importante elegir la imagen cuyas altas frecuencias se van a calcular (que será aquella que tenga originalmente los bordes más marcados) y la imagen cuyas bajas frecuencias se vayan a calcular. Para obtener un buen resultado, es crucial disponer de una función de convolución correctamente implementada, ya que esta se utiliza para extraer tanto las altas como las bajas frecuencias de las imágenes.

Este es un ejercicio delicado, donde cualquier desfase o desplazamiento producido en la imagen puede afectar negativamente a la imagen final. En la Figura 14.4 se muestra el resultado

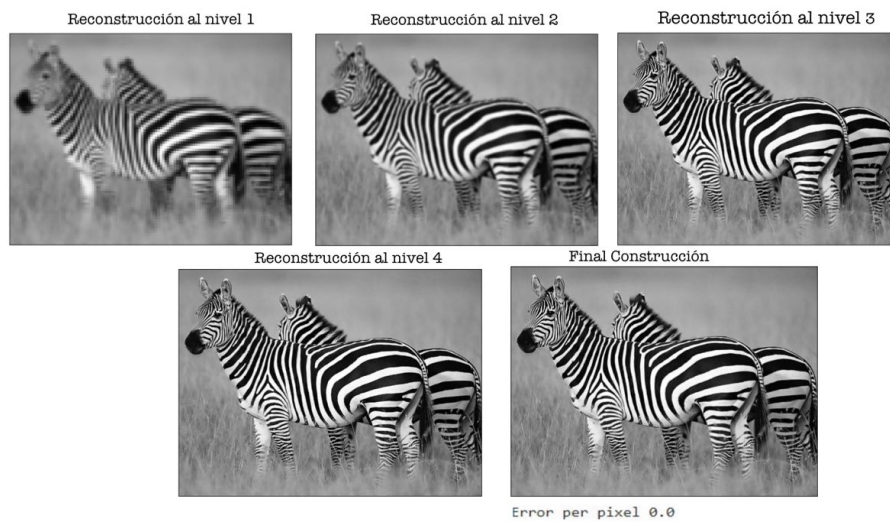


Figura 14.3.: Reconstrucción de la imagen original a partir de la Pirámide Laplaciana.

obtenido para un par de imágenes. Como se puede observar, en la imagen más grande de la pirámide se percibe claramente la figura de Einstein. Sin embargo, a medida que se asciende en la pirámide y la imagen se reduce de tamaño, comienza a aparecer la imagen de Marilyn Monroe.

14.2.2. Experimentación con el Algoritmo Propuesto. Segunda Parte

Se ha utilizado el método de convolución en los problemas anteriores de manera exitosa. A continuación se realiza una comparación de varios métodos de convolución utilizando diferentes librerías y enfoques (NumPy, SciPy, OpenCV, FFT con NumPy y FFT con TensorFlow) sobre datos sintéticos de distintos tamaños. El resultado obtenido usando cada algoritmo es el mismo y el objetivo reside en medir y comparar el tiempo de ejecución de cada método para diferentes configuraciones.

Los métodos que se van a comparar son los siguientes, algunos de los cuales son ampliamente utilizados en VC, especialmente la versión de OpenCV por su implementación altamente optimizada.

- **Convolución usando NumPy:** Realiza la convolución utilizando bucles for y operaciones de suma y multiplicación. Es una implementación siguiendo el algoritmo a fuerza bruta usando la librería Numpy.
- **Convolución usando SciPy:** Utiliza la función `convolve2d` de SciPy para realizar la convolución. Es una implementación más optimizada que la anterior.
- **Convolución usando OpenCV:** Utiliza la función `filter2D` de OpenCV para realizar la convolución. Es una de las implementaciones más rápidas ya que usa programación de bajo nivel, y está altamente optimizada. Usa la GPU para mejorar el rendimiento.

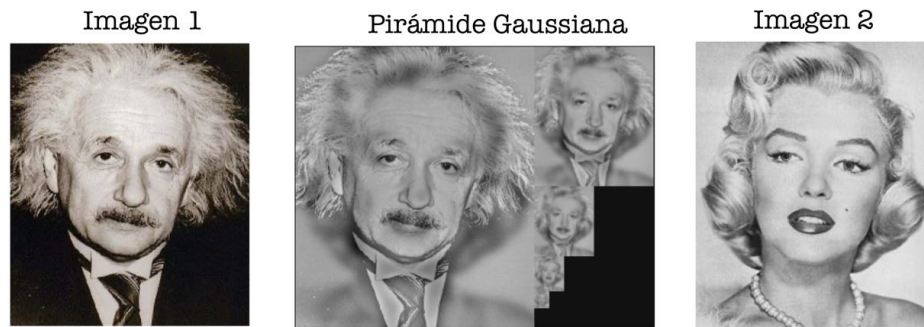


Figura 14.4.: Ejemplo de imagen híbrida a partir de las imágenes situadas en los extremos.

- **Convolución usando el Algoritmo `convolucion_fft`:** Función propia que utiliza la FFT para realizar la convolución en el dominio de la frecuencia.
- **Convolución usando el Algoritmo `convolucion_fft` ejecutado sobre GPU:** Similar al método anterior, pero utilizando TensorFlow para aprovechar las optimizaciones y la GPU.

En detalle, se presenta el tiempo de ejecución de cada método para distintos valores de tamaño del kernel: 3, 5, 7, 9, 11, 16, 20 y 24. Estos valores se analizan en el contexto de diferentes tamaños de imágenes, específicamente 128, 256, 512, 1024, 2048, 3000, 4000 y 5000 píxeles. Para cada combinación de tamaño del kernel y tamaño de la imagen, se mide y se muestra el tiempo que cada método tarda en ejecutarse.

14.2.2.1. Resultados

Los resultados se pueden visualizar en la Figura 14.5. Analizando estos, es evidente que el método implementado mediante Numpy muestra una falta de eficiencia incluso para tamaños de kernel e imagen relativamente pequeños.

En general, el método que hemos propuesto y que está implementado para aprovechar la capacidad de la GPU muestra una superioridad notable sobre la mayoría de los algoritmos convencionales, con la excepción notable del algoritmo utilizado en OpenCV. Específicamente, para kernels de tamaño más pequeño, OpenCV tiende a ofrecer tiempos de procesamiento más reducidos en comparación con nuestro método propuesto.

No obstante, a medida que incrementamos las dimensiones del kernel, la disparidad en el rendimiento se hace más notable. En particular, cuando se utiliza un kernel de tamaño 9, ya empezamos a observar una mejora significativa en los tiempos de procesamiento con el método que hemos propuesto para la GPU. Esta tendencia se acentúa aún más con un kernel de tamaño 11, donde nuestro método comienza a demostrar una superioridad clara en términos de eficiencia.

Este fenómeno es particularmente interesante considerando que nuestra implementación, a pesar

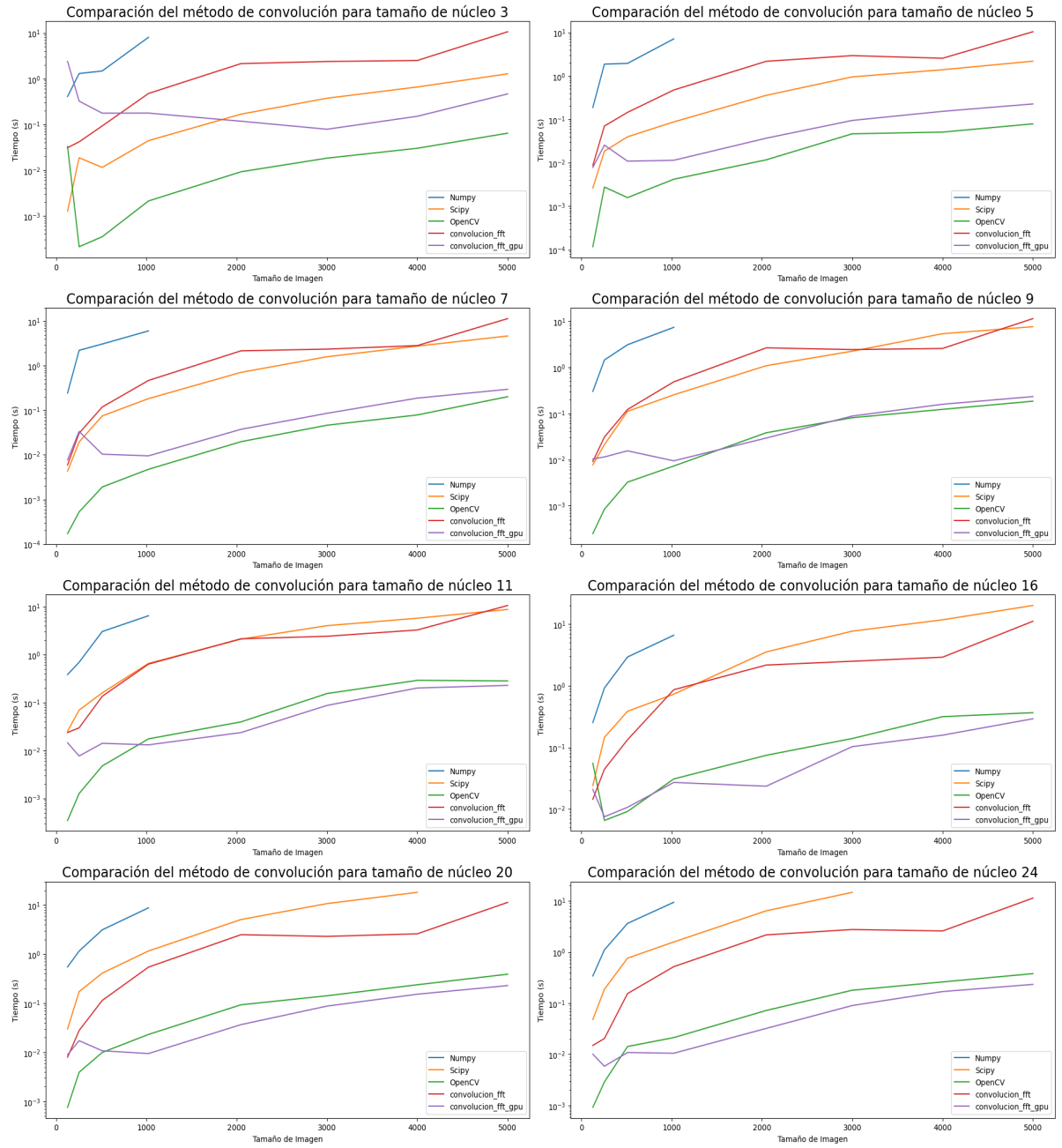


Figura 14.5.: Comparación del rendimiento según los valores del núcleo y del tamaño de imagen.

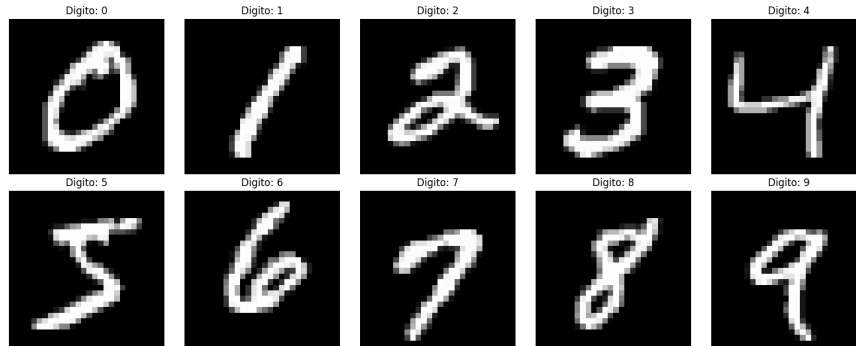


Figura 14.6.: Ejemplo del conjunto de datos MNIST.

de estar diseñada para aprovechar la aceleración por GPU, no está completamente optimizada ni emplea técnicas de programación paralela en su totalidad. Esto constata que incluso una implementación no completamente refinada del método propuesto supera a las técnicas usadas comunmente en el AA. Este análisis coincide con el que se ha realizado de manera teórica.

14.3. Entrenamiento de CNN en el Dominio de la Frecuencia

Se integrará el método de convolución propuesto en la arquitectura de CNN, tal como se ha planteado en la sección teórica. Posteriormente, se realizará una comparación exhaustiva entre esta nueva arquitectura y una arquitectura clásica de CNN, abordando dos problemas distintos de clasificación en el ámbito del DL.

14.3.1. Conjunto de Datos

14.3.1.1. MNIST

El dataset MNIST [27] es uno de los conjuntos de datos más utilizados por los usuarios que comienzan a realizar pruebas con herramientas de AA debido a la facilidad de comprensión de los problemas que ofrece su uso. MNIST es un subconjunto de dígitos escritos a mano basado en otro conjunto más grande, llamado NIST [74]. Consta de 70,000 imágenes, 60,000 para entrenamiento y 10,000 para test. Las imágenes de la base de datos NIST están en blanco y negro y tienen un tamaño de 20x20 píxeles, pero el subconjunto de imágenes de MNIST se normalizó y se suavizó, lo que provocó que pasaran a ser de 28x28 píxeles y se introdujesen niveles de escala de grises. Las imágenes se dividen en 10 categorías, correspondientes a los números del 0 al 9, escritos a mano alzada.

El uso de MNIST ha sido fundamental en la investigación y desarrollo de técnicas avanzadas de DL, como las CNN. A lo largo de los años, ha servido como un estándar de referencia para evaluar y comparar el rendimiento de nuevos métodos y arquitecturas en tareas de reconocimiento de patrones y clasificación. Véase un ejemplo de este conjunto de datos en la Figura 14.6.

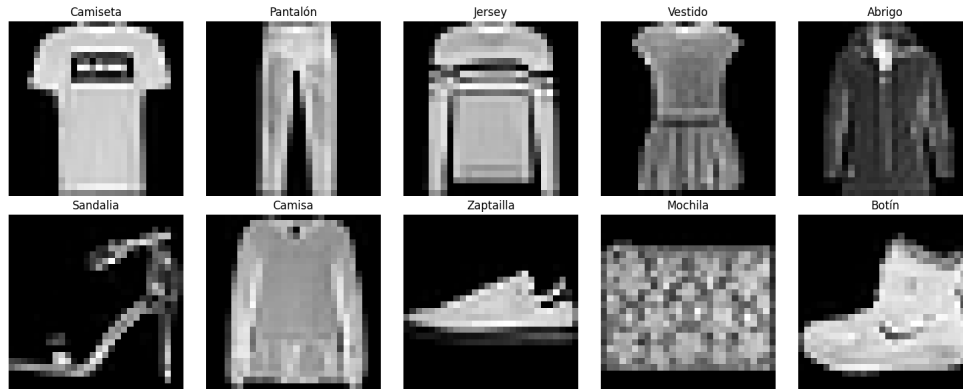


Figura 14.7.: Ejemplo del conjunto de datos Fashion MNIST.

14.3.1.2. Fashion-MNIST

Las imágenes del conjunto de datos Fashion-MNIST [28] corresponden a artículos de Zalando. Fashion MNIST consta de 60,000 imágenes de entrenamiento y 10,000 para realizar pruebas. Cada imagen tiene una etiqueta que puede ser de 10 clases distintas, siendo estas: camiseta, pantalón, jersey, vestido, abrigo, sandalia, camisa, zapatilla, bolso y bota. Las imágenes son de tamaño 28x28 y están en escala de grises, siguiendo el mismo patrón que el conjunto MNIST. Esto es porque se pretende que Fashion MNIST sea el reemplazo de MNIST, ya que este último dataset es muy simple.

Este conjunto de datos es utilizado extensivamente en el ámbito del AA y la VC, especialmente para la tarea de clasificación de imágenes. Su simplicidad y tamaño manejable lo convierten en una excelente opción para quienes están aprendiendo y desarrollando modelos de redes neuronales. Al igual que MNIST, permite a los investigadores y practicantes comparar sus resultados de manera efectiva con los de otros trabajos previos. Además, debido a la diversidad de las clases de ropa, Fashion MNIST presenta un desafío más complejo y realista en comparación con MNIST, lo que ayuda a los investigadores a desarrollar y evaluar modelos más robustos. Véase un ejemplo de este conjunto de datos en la Figura 14.7.

14.3.2. Protocolo de Validación

Se reserva un 20 % de los datos de entrenamiento para validación. En este conjunto de datos de validación se mide el rendimiento del modelo durante el proceso de entrenamiento. La validación ayuda a ajustar los hiperparámetros del modelo y a prevenir el sobreajuste. Esto asegura que el modelo generalice bien a datos nuevos e invisibles, mejorando su capacidad para hacer predicciones precisas en situaciones del mundo real.

14.3.3. Función de Pérdida

La función de pérdida guía el proceso de entrenamiento del modelo, indicando cómo ajustar los pesos del mismo para minimizar el error de predicción.

Se usará la entropía cruzada, que es una función de pérdida utilizada principalmente en

problemas de clasificación multiclase. Mide la diferencia entre la distribución de probabilidad verdadera y la distribución de probabilidad predicha por el modelo. La fórmula es la siguiente:

$$L = - \sum_{i=1}^N \sum_{c=1}^C y_{i,c} \log(\hat{y}_{i,c})$$

donde:

- N es el número de muestras.
- C es el número de clases.
- $y_{i,c}$ es la etiqueta verdadera (1 si la muestra i pertenece a la clase c , de lo contrario 0).
- $\hat{y}_{i,c}$ es la probabilidad predicha de la muestra i para la clase c .

14.3.4. Métricas de rendimiento

Para medir el rendimiento de un modelo de clasificación, es fundamental presentar ciertos conceptos básicos que describen los diferentes tipos de resultados de predicción.

- **Verdaderos Positivos (TP).** Las instancias correctamente clasificadas como positivas.
- **Falsos Positivos (FP).** Las instancias incorrectamente clasificadas como positivas.
- **Verdaderos Negativos (TN).** Las instancias correctamente clasificadas como negativas.
- **Falsos Negativos (FN).** Las instancias incorrectamente clasificadas como negativas.

En base a estos se pueden definir diversas métricas evaluación para medir el rendimiento del modelo. Cada métrica proporciona información específica sobre diferentes aspectos. A continuación se describen algunas de las métricas más comunes para un problema de clasificación.

- **Accuracy.** Es una métrica que mide la proporción de predicciones correctas realizadas por un modelo sobre el total de predicciones realizadas. Matemáticamente, se calcula como la suma de predicciones correctas dividida por el número total de predicciones. Sin embargo, esta métrica puede ser engañosa si hay un desequilibrio entre las clases objetivo, ya que puede dar una impresión falsa de la calidad del modelo.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Precisión.** La precisión mide la proporción de verdaderos positivos entre todas las predicciones positivas.

$$\text{Precisión} = \frac{TP}{TP + FP}$$

- **Exhaustividad.** La exhaustividad mide la proporción de verdaderos positivos entre todas las muestras que son realmente positivas.

$$\text{Exhaustividad} = \frac{TP}{TP + FN}$$

■ **F_1 -Score.**

El F1-Score es una métrica que combina precisión y recuperación en una sola medida. Es especialmente útil cuando hay un desequilibrio entre las clases objetivo, ya que tiene en cuenta tanto los falsos positivos como los falsos negativos. El F1-Score se calcula como la media armónica de precisión y recuperación.

$$F1 = 2 \cdot \frac{\text{Precisión} \cdot \text{Sensibilidad}}{\text{Precisión} + \text{Sensibilidad}}$$

Por último, se empleará el tiempo, como métrica, dado que el propósito de adoptar una implementación alternativa tiene como objetivo específicamente estudiar el aceleramiento en el proceso de entrenamiento de las CNN.

14.3.5. Arquitectura DL de CNN

Se abordarán ambos problemas de clasificación, MNIST y Fashion-MNIST, usando la arquitectura de DL que se muestra en la Tabla 14.1.

Existen otras arquitecturas más complejas que pueden ser empleadas para resolver este problema y que, potencialmente, obtienen mejores resultados en términos de precisión y capacidad predictiva. Estas por lo general al ser más complejas llevan a tiempos de entrenamiento más altos.

En este estudio, nuestro objetivo no es identificar el modelo que ofrezca el mejor rendimiento en términos de capacidad de clasificación, sino comparar una arquitectura concreta con una “equivalente” en el dominio de la frecuencia para evaluar si se pueden lograr mejoras significativas en términos de eficiencia temporal sin comprometer los resultados de manera significativa.

Capa	Tamaño del Núcleo	Dimensión de Entrada	Salida	Canales de Entrada	Salida
Conv	3×3	28×28	28×28	1	32
ReLU	-	28×28	28×28	-	-
MaxPooling	2×2	28×28	14×14	-	-
Conv	5×5	14×14	10×10	32	16
ReLU	-	10×10	10×10	-	-
MaxPooling	2×2	10×10	5×5	-	-
Flatten	-	400	400	-	-
FC	-	400	100	-	-
ReLU	-	100	100	-	-
FC	-	100	10	-	-

Tabla 14.1.: Arquitectura de la CNN modificada.

Es importante señalar que no se utilizarán técnicas de regularización a parte de la parada temprana, ya que el objetivo es comparar una arquitectura sencilla con la implementación que se realizará en el dominio de la frecuencia, la cual no incorpora estas técnicas.

14.3.5.1. Entrenamiento de MNIST

Se realiza el entrenamiento del modelo descrito en la Tabla 14.1 utilizando el conjunto de datos MNIST. La función de pérdida y las métricas que se emplearán ya han sido previamente

14.3. ENTRENAMIENTO DE CNN EN EL DOMINIO DE LA FRECUENCIA

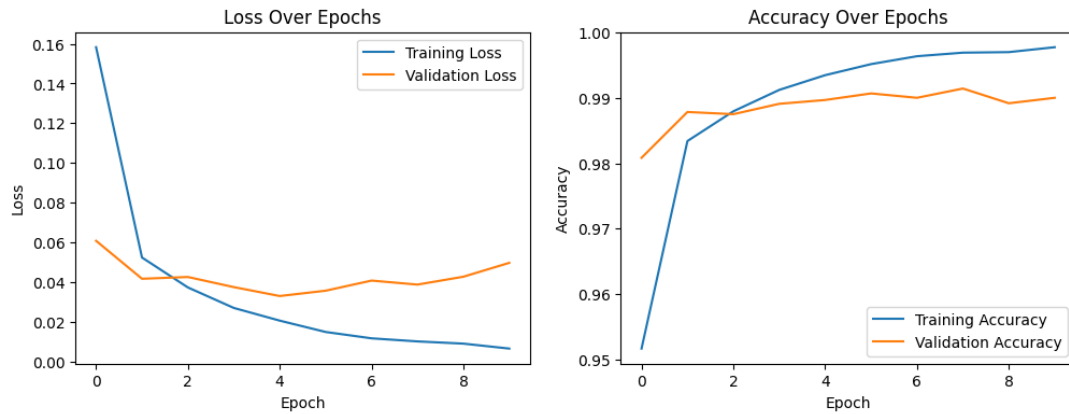


Figura 14.8.: Curva de Pérdida y de métricas.

definidas. Se ha ejecutado un grid search para determinar los **hiperparámetros** a utilizar:

- Tamaño de batch: 16.
- Optimizador Adam con una tasa de aprendizaje inicial de 0.0006562.
- Criterio de parada temprana.
- Número de épocas: 10.

Los resultados que se obtienen en el entrenamiento se muestran descritos en la Tabla 14.2.

Conjunto de Datos	Pérdida	Accuracy	F1-Score
Entrenamiento	0.0065	0.9977	0.9977
Validación	0.0497	0.9900	0.9899
Tiempo de Entrenamiento: 176 segundos			

Tabla 14.2.: Resultados del conjunto de datos de entrenamiento y validación.

Si se analiza el conjunto de validación, se observa que tanto el accuracy como el F1-score tienen valores altos, lo cual indica que el modelo está realizando predicciones precisas y equilibradas. En cuanto al tiempo de entrenamiento, este ha sido de aproximadamente **3 minutos**, lo cual es un tiempo más que aceptable dada la simplicidad del problema.

En la Figura 14.8, se presentan las curvas de aprendizaje y las métricas clave, proporcionando una visión detallada del proceso de aprendizaje del modelo. A partir de la décima época, se observa que el modelo comienza a sobreajustarse. Esto se evidencia por el aumento del error en el conjunto de validación a partir de ese punto.

Se aplica ahora el modelo al conjunto de Test. Los resultados se muestran en la Tabla 14.3. Estos resultados están en línea con lo observado durante la validación, siendo muy satisfactorios ya que se supera el 98 % en ambas métricas.

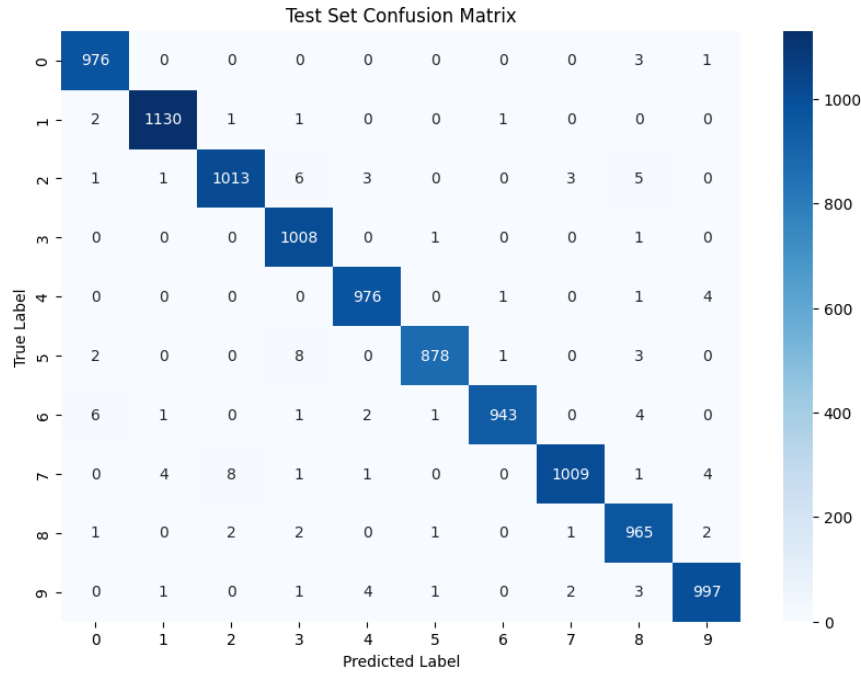


Figura 14.9.: Matriz de Confusión del conjunto de Test.

Conjunto de Datos	Accuracy	F1-Score
Test	0.9895	0.9895

Tabla 14.3.: Resultados de test.

Si se atiende a la matriz de confusión en la Figura 14.9, correspondiente al conjunto de prueba, esta permite identificar y analizar los errores más comunes cometidos por el modelo. La matriz de confusión es una herramienta esencial para comprender en detalle el rendimiento del modelo, ya que muestra no solo la cantidad de aciertos, sino también los tipos específicos de errores que se han producido.

Cada celda de la matriz indica el número de predicciones realizadas por el modelo para cada clase, proporcionando una visión clara de cuántos ejemplos fueron correctamente clasificados y cuántos fueron mal clasificados en cada categoría. Este análisis detallado ayuda a identificar patrones en los errores, como clases que el modelo tiende a confundir con mayor frecuencia, y proporciona una base sólida para implementar mejoras y ajustes en el modelo. Como era de esperar, la mayor debilidad del modelo se encuentra en la diferenciación de los dígitos que tienen trazos similares. Específicamente, los dígitos 4 y 9, 2 y 7, así como 6 y 0, son los más propensos a ser confundidos. Estas combinaciones presentan un desafío particular debido a sus características visuales compartidas.

Por último, en la Figura 14.10 se muestran algunas de las imágenes que el modelo ha clasificado incorrectamente. Esta visualización permite analizar los errores más comunes y comprender mejor las limitaciones del modelo. Esta información ya había sido identificada mediante la matriz

14.3. ENTRENAMIENTO DE CNN EN EL DOMINIO DE LA FRECUENCIA

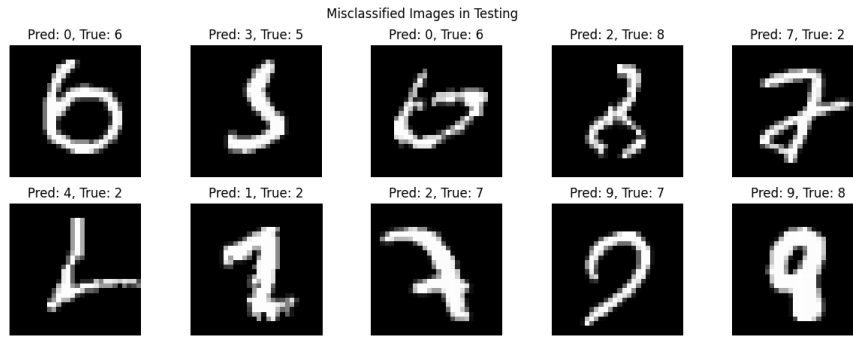


Figura 14.10.: Ejemplos de imágenes clasificadas erróneamente en el conjunto de Test.

de confusión. Sin embargo, puede resultar interesante visualizar estos casos para determinar si realmente existe algún tipo de ambigüedad o para comprender mejor el origen de la confusión. Se muestra, por ejemplo, en la tercera imagen de la primera fila de la Figura 14.10, que clasificar la imagen es complicado hasta para un experto por la similitud del trazo tanto con el dígito 6 como el dígito 0.

La tarea de clasificación de dígitos del conjunto de datos MNIST es relativamente sencilla y, con la CNN seleccionada, se obtienen resultados considerablemente buenos.

14.3.5.2. Entrenamiento de Fashion-MNIST

Se realiza el entrenamiento del modelo descrito en la Tabla 14.1 utilizando el conjunto de datos Fashion-MNIST como hicimos con el conjunto MNIST. La función de pérdida y las métricas que se emplearán ya han sido previamente definidas. A continuación se ha ejecutado un grid search para determinar los **hiperparámetros** a utilizar:

- Tamaño de Batch: 16.
- Optimizador Adam con una tasa de aprendizaje inicial de 0.000656253.
- Criterio de parada temprana.
- Número de épocas: 13.

Los resultados que se obtienen en el entrenamiento se ven descritos en la Tabla 14.4.

Conjunto de Datos	Pérdida	Accuracy	F1-Score
Entrenamiento	0.0953	0.9647	0.9647
Validación	0.2833	0.9159	0.9158
Tiempo de Entrenamiento: 200 segundos			

Tabla 14.4.: Resultados del conjunto de datos de entrenamiento y validación.

Al observar el conjunto de validación, que se utiliza para evaluar el rendimiento del modelo, se puede comprobar que ambas métricas proporciona resultados buenos para un modelo tan sencillo. Estos resultados son peores que los obtenidos para el dataset MNIST, lo cual refuerza



Figura 14.11.: Curva de Pérdida y de métricas.

la idea de que es un problema de clasificación más complejo.

En cuanto al tiempo de entrenamiento, este es de aproximadamente **3 minutos y 20 segundos**, lo cual es un tiempo aceptable. Esto se debe a que el número de épocas ejecutadas no es muy alto y el conjunto de datos no es excesivamente grande. Mediante el uso de modelos más complejos, es posible alcanzar resultados superiores, aunque esto conlleva un aumento en el tiempo de entrenamiento.

En la Figura 14.11 se presentan las curvas de aprendizaje y las métricas clave, proporcionando una visión detallada del proceso de aprendizaje del modelo.

Se aplica ahora el modelo al conjunto de Test. Estos resultados se muestran en la Tabla 14.5. En efecto, estos son similares a los obtenidos en el conjunto de validación en el proceso de entrenamiento, salvo porque en este caso la métrica de F1Score es algo superior a la de Accuracy. En general es un resultado bueno dada la simplicidad del modelo.

Conjunto de Datos	Accuracy	F1-Score
Test	0.9117	0.9210

Tabla 14.5.: Resultados de test

A continuación, se presenta la confusión en la Figura 14.12. Como era de esperar, la mayor debilidad del modelo se encuentra en diferenciar tipos de calzado o partes superiores entre sí como camisa y camiseta por las características comunes que comparten. Esto se puede comprobar al observar la matriz de confusión, donde se aprecia que la clase 0 (camisa) y la clase 6 (camiseta) tienen el mayor número de equivocaciones. Además, la clase 6 (camiseta) también se confunde frecuentemente con la clase 4 (abrigo). Por otro lado, la clase 5 (sandalias) se confunde únicamente con las otras dos clases de zapatos.

Por último, se ilustra en la Figura 14.13 algunas de las imágenes del conjunto de Test erróneamente clasificadas. Nótese por ejemplo, en la segunda imagen de la primera fila de la Figura 14.13, una bota ha sido incorrectamente clasificada como sandalia, aunque ambos sean tipos de

14.3. ENTRENAMIENTO DE CNN EN EL DOMINIO DE LA FRECUENCIA

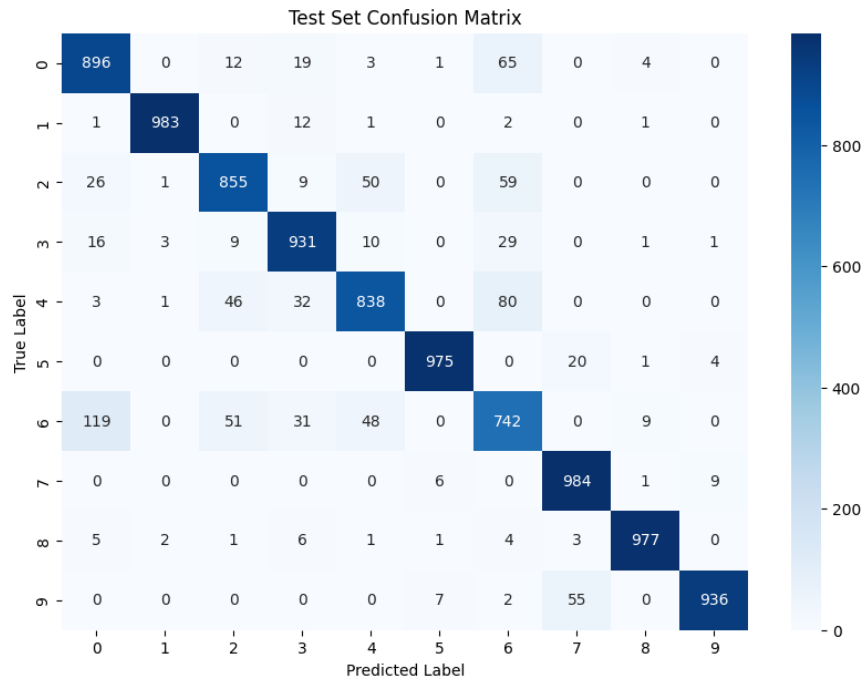


Figura 14.12.: Matriz de Confusión del conjunto de Test. 0: 'T-shirt', 1: 'Trouser', 2: 'Pullover', 3: 'Dress', 4: 'Coat', 5: 'Sandal', 6: 'T-Shirt', 7: 'Sneaker', 8: 'Bag', 9: 'A.boot'.

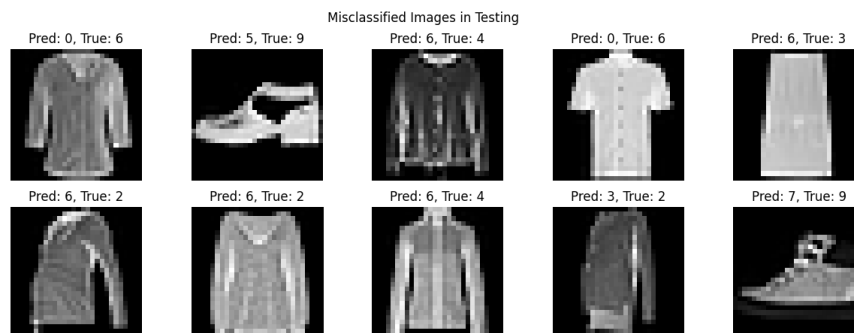


Figura 14.13.: Ejemplos de imágenes erróneamente clasificadas en el conjunto de Test.

calzado. De forma similar, se muestra también la confusión entre una camisa y una camiseta son ambas prendas de vestir superiores, y sin características distintivas claras en las imágenes, el modelo podría confundirlas.

La tarea de clasificación de prendas de ropa utilizando el dataset Fashion-MNIST es más compleja en comparación con la clasificación de dígitos en el dataset MNIST. Esta mayor complejidad se traduce en resultados generalmente inferiores al usar el mismo modelo de clasificación.

14.3.6. Modelo Alternativo de la CNN

Se quiere ejecutar el entrenamiento de una CNN en el dominio de la frecuencia, con el objetivo de evaluar su viabilidad y rendimiento. Para lograr esto, se ha desarrollado una implementación personalizada que realiza dicho entrenamiento para una arquitectura similar a la mostrada en la Tabla 14.1. Esta implementación es un primer paso dentro de la línea de investigación propuesta en este trabajo. Se centra en aplicar la FFT a las entradas y realizar las operaciones convolucionales en el dominio de la frecuencia, lo que podría potencialmente reducir el costo computacional y mejorar la eficiencia.

Adaptar todo el proceso de entrenamiento a este nuevo dominio ha sido un desafío significativo y ha requerido una considerable cantidad de tiempo y esfuerzo. La transformación de datos, la redefinición de operaciones convolucionales y la integración con las técnicas de aprendizaje profundo existentes han presentado múltiples obstáculos técnicos.

Entre otras tareas, ha sido necesario redefinir algunas de las capas que componen la arquitectura de una CNN para que puedan operar con números complejos.

14.3.6.1. Función de Activación

Se define la siguiente función no lineal denominada ReLUCompleja $f : \mathbb{C} \rightarrow \mathbb{C}$. Esta se define como:

$$f(z) = \begin{cases} z & \text{si } |z| > \delta, \\ 0 & \text{si } |z| \leq \delta, \end{cases}$$

donde $z \in \mathbb{C}$ es un número complejo igual a $a + ib$, $|z| = \sqrt{a^2 + b^2}$, 0 es el punto (0,0) en el plano complejo y δ es un umbral ajustable de este método. Esta solución se puede considerar como un equivalente de la función ReLU tradicional para números complejos. Se toma $\delta = 0.1$ para la implementación.

14.3.6.2. Capa Convolutiva

La primera y principal parte de una red neuronal convolutiva es la propia convolución, donde multiplicamos elemento a elemento las imágenes (o series temporales) por los valores adecuados de los núcleos convolucionales, los cuales han sido previamente transformados al dominio de la frecuencia. Si utilizamos núcleos más pequeños que el tamaño de las imágenes o la longitud de las series temporales, es necesario aplicar padding con ceros a los núcleos antes de la transformación para la multiplicación punto a punto. Gracias a este padding, después de la transformación,

siempre realizamos las operaciones de multiplicación con matrices del mismo tamaño. Así, podemos ahorrar aún más operaciones si el tamaño del núcleo es mayor. Sin embargo, dado que toda nuestra red funciona en el dominio de Fourier, he aplicado otro enfoque: generamos los núcleos directamente en el dominio de la frecuencia en lugar de transformarlos desde el dominio del tiempo mediante la transformada de Fourier. De esta forma, ahorramos el costo de la transformación de los núcleos durante el entrenamiento. En este caso, el tamaño del núcleo es el mismo que el tamaño de la entrada. De esta manera, se realizan potencialmente menos operaciones que en una arquitectura clásica, ya que solo se deben gestionar las transformadas de la imagen y el producto puntual.

14.3.6.3. Capa Pooling

Empleamos el método de pooling espectral presentado en [76] como una técnica de submuestreo. En este método, la reducción de dimensionalidad se lleva a cabo en el dominio de Fourier, donde la matriz de entrada de $N \times M$ se recorta, conservando solo la submatriz central de frecuencias de $H \times W$. Este enfoque se distingue de otras estrategias de pooling en el dominio del tiempo, como el max pooling, que disminuye la dimensionalidad al menos en un factor de 4 en situaciones bidimensionales, y donde el valor agregado en cada ventana no siempre representa adecuadamente el contenido de dicha ventana. Por el contrario, el pooling espectral permite ajustar la dimensionalidad de la salida y puede también considerarse un filtro, dado que las frecuencias más altas eliminadas en el proceso representan el ruido en el caso bidimensional.

14.3.6.4. Capa Flatten

En los análisis teóricos se ha planteado que en la última capa se realice una transformada inversa y se vuelva al dominio espacial, esto es lo que proponía los trabajos futuros de [10]. Sin embargo, he optado por realizar un enfoque alternativo para reducir el número de operaciones. Antes de aplanar el mapa de características de la última capa de convolución, calculamos la magnitud de los valores complejos aplicando una función $f : \mathbb{C} \rightarrow \mathbb{R}$, que se puede escribir de la siguiente manera:

$$f(a + ib) = a^2 + b^2 \quad \forall (a + ib) \in \mathbb{C}.$$

Nótese que la complejidad computacional de este cálculo es $O(n)$, en lugar de $O(n \log(n))$ de la FFT inversa. Después de la capa de flatten, se utiliza una red neuronal totalmente conectada tradicional con una sola capa para predecir las clases.

Esta implementación de una CNN realiza, por tanto, un número considerablemente menor de operaciones. Esto se debe a que no solo reutiliza la DFT para las convoluciones intermedias, sino que también evita la necesidad de realizar transformadas adicionales para cada kernel. Además, la implementación omite la IDFT al final, regresando al dominio real mediante una capa de magnitud, lo que reduce aún más la carga computacional. Esto podría no reflejarse en el tiempo de entrenamiento, ya que existen otros factores involucrados, como la optimización de las implementaciones, la utilización eficiente del hardware, la paralelización de procesos, entre otros.

A continuación, se examina la viabilidad del nuevo modelo y los resultados obtenidos para los dos conjuntos de datos: MNIST y Fashion-MNIST.

14.3.6.5. Entrenamiento de MNIST

Para la selección de hiperparámetros, se llevó a cabo un grid search manual, y finalmente se establecieron los siguientes parámetros:

- Tamaño de Batch: 60.
- Número de épocas: 10.
- Optimizador Adam con un learning Rate: 0.001.

En la Tabla 14.6, se muestran los resultados obtenidos, así como el tiempo que duró el entrenamiento.

Conjunto de Datos	Pérdida	Accuracy	F1-Score
Entrenamiento	0.0791	0.9745	0.9779
Validación	0.1054	0.9707	0.9680
Tiempo de Entrenamiento: 160 segundos			

Tabla 14.6.: Resultados del conjunto de datos de entrenamiento y validación.

Se observa una leve reducción en el tiempo de entrenamiento en comparación con la arquitectura anterior. Sin embargo, al analizar los resultados en el conjunto de validación, se aprecia una disminución en el rendimiento, con una reducción de aproximadamente el 2 % en las métricas de evaluación. Lo que sugiere que, aunque es más eficiente en términos computacionales al realizar un menor número de operaciones, sacrifica cierta capacidad predictiva.

Se muestra en la Figura 14.14 las curvas de aprendizaje y la métrica de rendimiento de accuracy. En efecto, lo primero que se observa es que existe un aprendizaje del modelo, ya que las curvas de pérdida disminuyen tanto en entrenamiento como en validación, lo cual indica que el modelo planteado está aprendiendo. A partir de la época 10, se ha estudiado un comportamiento de sobreajuste, que indica que los resultados en validación comienzan a empeorar a partir de este valor. La curva de pérdida de entrenamiento comienza en un valor alto, lo que podría indicar una mala inicialización de los pesos. Sin embargo, este fenómeno se ha observado con varias inicializaciones.

Evaluamos a continuación el modelo en el conjunto de prueba. Sin embargo, es importante destacar que la comparativa con el modelo anterior se realiza observando el conjunto de validación, de otra manera se estaría realizando Data Snooping. Esto es crucial, ya que comparar los resultados basándonos en el conjunto de prueba podría introducir sesgos y sobreestimaciones en el rendimiento del modelo. Los resultados se recogen en la Tabla 14.7.

Conjunto de Datos	Accuracy	F1-Score
Test	0.9636	0.9633

Tabla 14.7.: Resultados de test

Los resultados obtenidos son inferiores en comparación con los que otros modelos más sencillos pueden ofrecer para este problema.

Finalmente, en la Figura 14.15 se muestran algunas imágenes erróneamente clasificadas. Los

14.3. ENTRENAMIENTO DE CNN EN EL DOMINIO DE LA FRECUENCIA

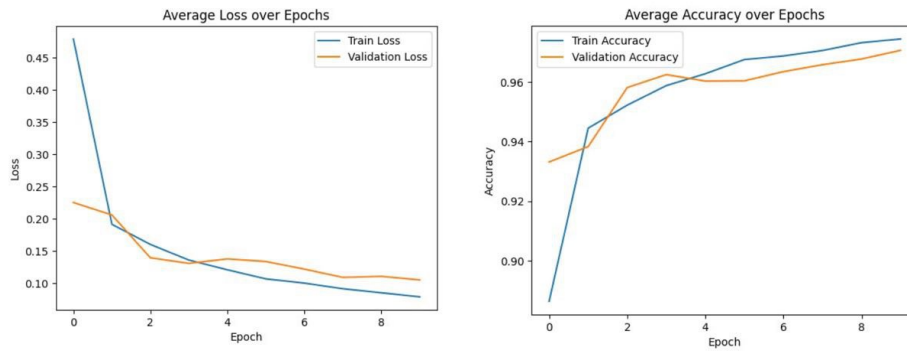


Figura 14.14.: Curvas de pérdida y de accuracy.

fallos que encontramos son mayormente aquellos relacionados con dígitos con trazos confusos o trazos que son muy similares entre ellos.

14.3.6.6. Entrenamiento de Fashion-MNIST

Para la selección de hiperparámetros, se llevó a cabo un grid search manual, y finalmente se establecieron los siguientes parámetros:

- Tamaño de Batch: 60.
- Número de épocas: 12.
- Optimizador Adam con un learning Rate 0.001:

En la Tabla 14.8, se muestran los resultados obtenidos en el conjunto de entrenamiento y de validación, así como el tiempo que duró el entrenamiento.

Conjunto de Datos	Pérdida	Accuracy	F1-Score
Entrenamiento	0.2953	0.8897	0.8871
Validación	0.3689	0.8718	0.8600
Tiempo de Entrenamiento: 200 segundos			

Tabla 14.8.: Resultados del conjunto de datos de entrenamiento y validación.

Al observar nuevamente el conjunto de validación, se obtienen tiempos de entrenamiento similares. Sin embargo, existe una reducción de aproximadamente un 4% en el rendimiento, evaluado mediante las métricas de accuracy y F1-score. Se muestra en la Figura 14.16 las curvas de aprendizaje y la métrica de rendimiento de accuracy. En efecto, se sigue observando un aprendizaje del modelo, ya que las curvas de pérdida disminuyen tanto en entrenamiento como en validación. A partir de la época 12, se ha estudiado un comportamiento de sobreajuste, que indica que los resultados en validación comienzan a empeorar a partir de este valor. Se sigue observando una curva de pérdida de entrenamiento que comienza en un valor alto como ocurría con el dataset anterior.

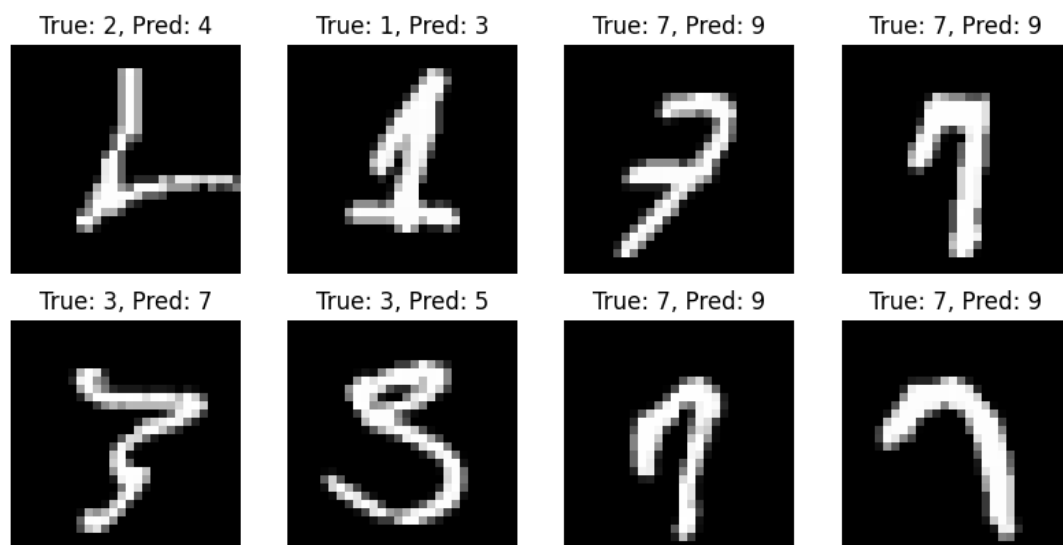


Figura 14.15.: Imágenes erróneamente clasificadas en el conjunto de Test.

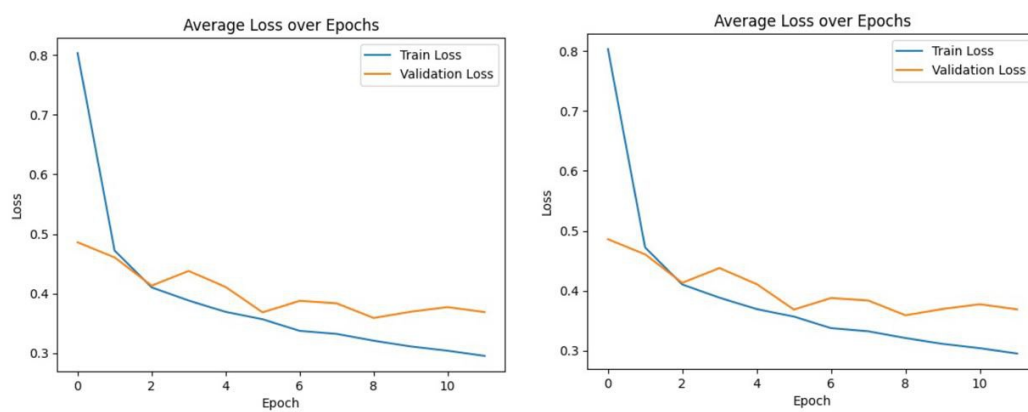


Figura 14.16.: Curvas de pérdida y de accuracy.

14.3. ENTRENAMIENTO DE CNN EN EL DOMINIO DE LA FRECUENCIA

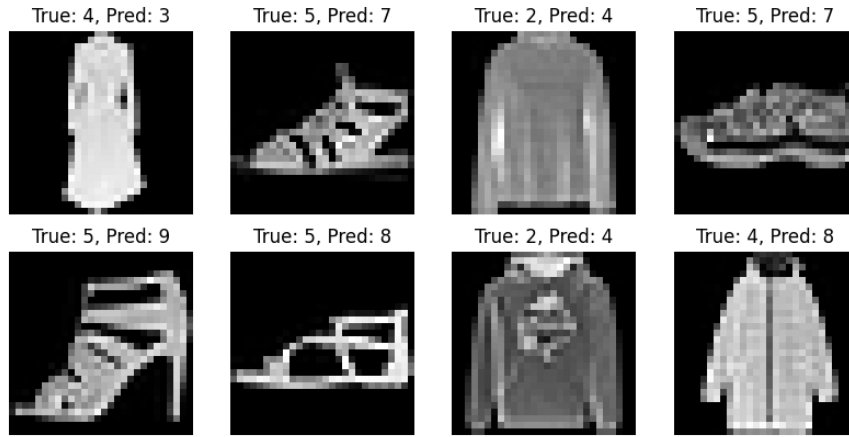


Figura 14.17.: Imágenes erróneamente clasificadas en el conjunto de Test. 0: 'T-shirt', 1: 'Trouser', 2: 'Pullover', 3: 'Dress', 4: 'Coat', 5: 'Sandal', 6: 'T-Shirt', 7: 'Sneaker', 8: 'Bag', 9: 'A.boot'.

De la misma manera evaluamos el modelo en el conjunto de Test. Los resultados se recogen en la Tabla 14.9.

Conjunto de Datos	Accuracy	F1-Score
Test	0.8372	0.8373

Tabla 14.9.: Resultados de test

Los resultados obtenidos en el conjunto de datos de prueba son significativamente inferiores a los que se pueden alcanzar con un modelo sencillo para la resolución de este problema.

Finalmente, en la Figura 14.17 se muestran algunas imágenes erróneamente clasificadas. Como se mencionó anteriormente, los errores que se observan son principalmente debidos a la confusión entre diferentes tipos de calzado o de prendas superiores por las características similares que guardan.

14.3.6.7. Comentarios Finales

En conclusión, el modelo planteado ha aprendido, y los resultados obtenidos con la red optimizada en el dominio de la frecuencia son aproximadamente un 2 % inferiores en el caso de MNIST y un 4 % en el caso de Fashion-MNIST, en comparación con los obtenidos en el conjunto de validación del modelo anterior. Los tiempos de entrenamiento son similares, debido también a la simplicidad de los problemas planteados. Sin embargo, este nuevo modelo realiza menos operaciones que una arquitectura clásica, lo que puede contribuir a la eficiencia en términos de procesamiento.

Esta implementación presenta ciertas limitaciones, como la dificultad para incorporar técnicas de regularización y la falta de flexibilidad en la capa convolucional en términos de padding y stride.

No obstante, este enfoque representa un avance significativo en la línea de investigación de la aceleración del entrenamiento en CNN. Aunque todavía existen áreas que pueden mejorarse para optimizar su rendimiento y flexibilidad, abre una importante línea de reflexión sobre cuánto se está dispuesto a sacrificar en términos de precisión de los resultados a cambio de una mayor eficiencia computacional en términos de número de operaciones. Esta consideración es crucial para encontrar un equilibrio adecuado entre rendimiento y eficiencia en aplicaciones prácticas.

Parte III.

Conclusiones y Trabajos Futuros

Capítulo 15.

Conclusiones y Trabajos Futuros

En la parte matemática se propuso el estudio de la Transformada de Fourier en $\mathcal{L}^1(\mathbb{R}^n)$ y de su inversa, que se llevó a cabo en detalle en la primera parte del presente trabajo.

Este trabajo de profundización en dicho espacio ha planteado el reto añadido de identificar las condiciones bajo las cuales ciertas propiedades y resultados, que son naturalmente ciertos en espacios como $\mathcal{S}^1(\mathbb{R}^n)$, siguen siendo válidos en un espacio más general como $\mathcal{L}^1(\mathbb{R}^n)$. Este reto se ha manifestado de forma significativa en las áreas de derivación y aplicación de la Transformada de Fourier, obligando a un examen meticuloso de cómo estas operaciones interactúan dentro de este espacio más general. Esto ha supuesto un desafío por la falta de contenido bibliográfico que abordaba estas cuestiones en este espacio, lo cual me ha llevado a reflexionar cuidadosamente sobre las hipótesis necesarias y cómo estas influían en los resultados que se buscaban obtener.

Por otro lado, se ha realizado una comparación entre el comportamiento de la Transformada de Fourier en el espacio $\mathcal{L}^1(\mathbb{R}^n)$ y su comportamiento en otros espacios como $\mathcal{L}^2(\mathbb{R}^n)$ y $\mathcal{S}^1(\mathbb{R}^n)$. Esta comparativa ha enriquecido nuestra comprensión sobre la Transformada, ilustrando cómo se manifiesta en diferentes marcos teóricos.

Inspirados por el comportamiento de la Transformada en los marcos descritos y las aplicaciones en problemas de VC del Teorema de Convolución descritas en la primera parte del trabajo, se ha realizado un análisis detallado sobre la DFT y su relación con el dominio de la frecuencia. Este conocimiento, combinado con el entendimiento del algoritmo de la FFT, ha facilitado la definición de un nuevo método para ejecutar la operación de convolución empleando el Teorema de Convolución.

Este método ha sido probado en problemas de VC, donde ha demostrado un desempeño satisfactorio. Además, se ha realizado un análisis comparativo en datos sintéticos con otros algoritmos presentes en la literatura, como la versión de convolución de las librerías OpenCV o SciPy, revelando su superioridad, especialmente cuando se incrementa la dimensión del kernel.

Finalmente, se ha diseñado una nueva arquitectura para una CNN que ejecuta el entrenamiento completamente en el dominio de la frecuencia. Se ha llevado a cabo tanto un análisis teórico como experimental de la eficiencia de esta arquitectura, comparándola con una arquitectura clásica. El modelo diseñado realiza menos operaciones que una arquitectura clásica pero sin embargo el rendimiento en términos de precisión del modelo disminuye alrededor de un 2 % en el conjunto de datos MNIST y de un 4 % en el de Fashion-MNIST. Por lo tanto, los resultados pueden verse comprometidos a cambio de un modelo que realice menos operaciones y pudiera ser más eficiente energéticamente.

Esta investigación me ha proporcionado una comprensión más profunda del funcionamiento de las CNN y me ha permitido explorar las bases matemáticas y las propiedades fundamentales que las respaldan. Además, el estudio realizado ha integrado y aportado coherencia a los dos grados académicos que he completado, familiarizándome con la consulta y análisis de literatura matemática relevante para estos temas. Esta habilidad, que considero crucial, será de invaluable ayuda en mi desarrollo profesional.

15.1. Objetivos Satisfechos

Todos los objetivos que se habían propuesto al comienzo de este trabajo se han visto realizados con éxito:

1. Se ha llevado a cabo un análisis exhaustivo de la **Transformada de Fourier** y su **inversa**, explorando sus propiedades y los resultados más significativos asociados, así como su interacción con la operación de convolución a través del **Teorema de Convolución**.
2. Se ha llevado a cabo un análisis exhaustivo de la **eficiencia** del nuevo método propuesto para efectuar la convolución, utilizando el Teorema de Convolución. Este enfoque ha necesitado un estudio avanzado de los elementos implicados en dicho teorema, como la **DFT**, así como una evaluación previa detallada de la eficiencia del algoritmo de la **FFT**.
3. Se ha efectuado una investigación exhaustiva sobre el **estado del arte** en el campo del AA, centrándose específicamente, en las CNN para obtener una comprensión profunda de su arquitectura.
4. Se ha desarrollado una modelización de una arquitectura alternativa para CNN, que realice el entrenamiento íntegramente en el **dominio de la frecuencia**. Esta arquitectura está diseñada para ejecutar menos operaciones durante el proceso de entrenamiento. Se ha realizado un análisis del **coste computacional** de este método de entrenamiento y se ha realizado una comparativa con el coste computacional de los métodos de entrenamiento más habituales para CNN.
5. Finalmente, se lanzó un **estudio experimental** donde se han diseñado una serie de pruebas para evaluar la capacidad del algoritmo de convolución desarrollado. Estas pruebas incluyen una primera parte de experimentos tanto de la aplicación del mismo para problemas reales como comparativos entre el algoritmo propuesto y otros algoritmos existentes para analizar sus eficiencias relativas. Posteriormente, existe una segunda parte donde se ha integrado el nuevo algoritmo en una CNN para compararla con una arquitectura clásica en conjuntos de datos ampliamente usados en el estado del arte.

15.2. Trabajos Futuros y Comentarios

Teniendo en cuenta los resultados obtenidos en esta investigación, las posibles direcciones para futuros trabajos podrían incluir:

- Explorar otras funciones de activación que operen con números complejos y que puedan integrarse como una capa dentro de la arquitectura propuesta.
- Estudiar la incorporación de técnicas de regularización en la CNN propuesta en el dominio

de la frecuencia.

- Desarrollar una implementación avanzada y eficiente que opere a un nivel de abstracción más bajo y que incorpore capacidades de paralelización para optimizar el rendimiento del proceso.
- Investigar otras arquitecturas potenciales que adopten la filosofía de entrenamiento en el dominio de la frecuencia, pero que no comprometan el rendimiento del modelo en términos de precisión y efectividad.
- Estudiar en profundidad la Teoría de Distribuciones y el análisis de Fourier dentro de este innovador marco teórico. Examinar el comportamiento de la Transformada de Fourier en este nuevo contexto podría descubrir características útiles que posteriormente podrían ser aplicadas en campos como el procesamiento de señales, la optimización de algoritmos y el desarrollo de nuevas técnicas en AA.

Para finalizar, la realización de este trabajo de TFG ha representado un desafío considerable. Proponer un nuevo avance en una línea de investigación en el campo del DL, inspirados por un desarrollo matemático en el área del Análisis de Fourier, implica enfrentarse a un problema real del que se dispone de poca literatura y ejemplos previos. Este proyecto me ha brindado, por tanto, la oportunidad de integrar todo el conocimiento adquirido durante mi carrera, que junto con nuevas habilidades características de un trabajo de iniciación a la investigación, me ha permitido proponer soluciones a los desafíos que he ido encontrando. A través de este proceso, he desarrollado una comprensión más profunda de cómo las teorías matemáticas pueden aplicarse de manera práctica para avanzar en el campo de la IA, abriendo nuevas vías de exploración y experimentación. Por lo tanto, este trabajo demuestra cómo la combinación de ambas disciplinas puede producir resultados fructíferos.

Bibliografía

- [1] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., USA, 2006.
- [2] Azriel Rosenfeld. Computer vision: basic principles. *Proc. IEEE*, 76(8):863–868, 1988.
- [3] David A. Forsyth and Jean Ponce. *Computer Vision - A Modern Approach, Second Edition*. Pitman, 2012.
- [4] Stuart J. Russell, Peter Norvig, and Ming-Wei Chang. *Artificial intelligence : a modern approach*. Persons Series in Artificial Intelligence. Pearson, Hoboken, NJ, 4^a ed., global edition edition, 2022.
- [5] Jose Cuartas. El concepto de la convolución en gráficos para comprender las convolutional neural networks CNN, 2021.
- [6] James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19:297–301, 1965.
- [7] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.
- [8] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229, 1959.
- [9] T.M. Mitchell. *Machine Learning*. McGraw-Hill International Editions. McGraw-Hill, 1997.
- [10] Michael Mathieu, Yann LeCun, and Mikael Henaff. Fast training of convolutional networks through ffts. *arXiv preprint arXiv:1312.5851*, 2014.
- [11] Sufian Ben Yacoub, Beat Fasel, and J. Luttin. Fast face detection using MLP and FFT. 2000.
- [12] A. Bosch, A. Zisserman, and X. Munoz. Representing shape with a spatial pyramid kernel. In *Proceedings of the ACM International Conference on Image and Video Retrieval*, 2007.
- [13] L. Fei-Fei, R. Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Conference or Journal Title*, 2004.
- [14] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, 2002.
- [15] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. *CVPR*, 2009.
- [16] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

Bibliografía

- [17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1106–1114, 2012.
- [18] Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. Green ai. *Commun. ACM*, 63(12):54–63, 2020.
- [19] C.-C. Jay Kuo and Azad M. Madni. Green learning: Introduction, examples and outlook. *Journal of Visual Communication and Image Representation*, 90:103685, 2023.
- [20] Komaravolu Chandrasekharan Salomon Bochner. *Fourier Transforms*. Princeton University Press, 1949.
- [21] Tristan Rivière and Alessandro Pigati. *Fourier Analysis in Function Space Theory*. Curso impartido en la ETH Zürich, 2023.
- [22] Loukas Grafakos. *Classical and modern Fourier analysis*. Springer, 2014.
- [23] David W. Kammler. *A First Course in Fourier Analysis*. Cambridge University Press, 2008.
- [24] Anders Vretblad. *Fourier Analysis and Its Applications*. Springer, 2003.
- [25] R.E. Bellman. *Adaptive Control Processes: A Guided Tour*. Princeton Legacy Library. Princeton University Press, 2015.
- [26] Will Koehrsen. Overfitting vs. underfitting: A complete example. *Towards Data Science*, 405, 2018.
- [27] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.
- [28] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.
- [29] Alejandro Antonio Torres Garcia. *Biosignal processing and classification using computational learning and intelligence : principles, algorithms, and applications*. Academic Press, London, England, 2022.
- [30] Osvaldo Simeone. A brief introduction to machine learning for engineers. *CoRR*, abs/1709.02840, 2017.
- [31] Lorenzo Ciampiconi, Adam Elwood, Marco Leonardi, Ashraf Mohamed, and Alessandro Rozza. A survey and taxonomy of loss functions in machine learning, 2023.
- [32] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv*, 2016.
- [33] J. Kiefer and J. Wolfowitz. Stochastic estimation of the maximum of a regression function. *Annals of Mathematical Statistics*, 23(3):462–466, 1952.
- [34] Leon Bottou, Frank E. Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *arXiv*, 1606.04838, 2016.
- [35] Jesús Castro-Infantes, José M. Manzano, and Francisco Torralbo. Conjugate plateau constructions in product spaces, 2022. Preprint. arXiv: 2203.13162 [math.DG].
- [36] Stanford University. Cs231n: Convolutional neural networks for visual recognition. <https://cs231n.github.io/>, 2023. Accedido: fecha de acceso.

- [37] Richard Szeliski. *Computer Vision: Algorithms and Applications*. Springer Cham, 2 edition, 2022.
- [38] Sourish Dey. Cnn application on structured data: Automated feature extraction. Towards Data Science on Medium, 2018.
- [39] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [40] Ralph Neuneier and Hans Georg Zimmermann. *How to Train Neural Networks*, pages 369–418. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.
- [41] Yao Ma and Jiliang Tang. *Deep Learning on Graphs*. Cambridge University Press, 2021.
- [42] Siddharth Sharma, Simone Sharma, and Anidhya Athaiya. Activation functions in neural networks. *International Journal of Engineering Applied Sciences and Technology*, 04:310–316, 2020.
- [43] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.
- [44] MathWorks. Redes neuronales convolucionales. <https://es.mathworks.com/discovery/convolutional-neural-network.html>, 2023. Accedido: 28 de mayo de 2024.
- [45] Xiaoqiang Zhao, Benben Tuo, and Yongyong Hui. Deep learning with cbam-based CNN for batch process quality prediction. *Measurement Science and Technology*, 34, 2023.
- [46] Andrew Ng. Dl - class 2 - week 2 notebook. Online, 2024. Notebook for Machine Learning and Deep Learning basic concepts and sample codes.
- [47] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In Yves Lechevallier and Gilbert Saporta, editors, *Proceedings of COMPSTAT'2010*, pages 177–186, Heidelberg, 2010. Physica-Verlag HD.
- [48] Pavithra Solai. convolutions and backpropagations". Medium, 2020. Accedido el [fecha de acceso].
- [49] Giorgio Roffo. *Ranking to Learn and Learning to Rank: On the Role of Ranking in Pattern Recognition Applications*. PhD thesis, 2017.
- [50] Neptune.ai. Early stopping with neptune. Neptune Blog, 2021.
- [51] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.
- [52] Malik Magdon-Ismael Abu-Mostafa, Yaser S. and Hsuan-Tien Lin. *Learning from Data: A Short Course*. AMLBook, 2012.
- [53] Emiel Por, Maaïke van Kooten, and Vanja Sarkovic. Nyquist–shannon sampling theorem. *Leiden University*, 1(1):5, 2019.
- [54] L Lévesque. Revisiting the nyquist criterion and aliasing in data analysis. *European Journal of Physics*, 22(2):127, 2001.
- [55] D. Forsyth, Y. Boykov, L. Davis, W. Freeman, M. Hebert, D. Kreigman, and P. Duygulu. Lecture 4: Linear filters. Lecture, University, 2018. Many slides by (or adapted from) D. Forsyth, Y. Boykov, L. Davis, W. Freeman, M. Hebert, D. Kreigman, P. Duygulu.
- [56] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

- [57] Convolution, edges, template matching. University of California, Berkeley, 2017. Accedido el [fecha de acceso].
- [58] Julius O. Smith. *Spectral Audio Signal Processing*. accessed <date>. online book, 2011 edition.
- [59] J. Dongarra and F. Sullivan. Guest editors introduction to the top 10 algorithms. *Computing in Science; Engineering*, 2(01):22–23, 2000.
- [60] Carl Friedrich Gauss. Nachlass, theoria interpolationis methodo nova tractata. *Unpublished manuscript*, 3:265–330.
- [61] F. Yates and Commonwealth Bureau of Soils. *The Design and Analysis of Factorial Experiments*. Technical communication / Commonwealth Bureau of Soils. Commonwealth Agricultural Bureaux, 1964.
- [62] N.L.J. Design and analysis of industrial experiments. *Journal of the Institute of Actuaries*, 80(3):432–434, 1954. Review of book by G.E.P. Box, W.G. Hunter, and J.S. Hunter, Design and Analysis of Experiments.
- [63] I. J. Good. The Interaction Algorithm and Practical Fourier Analysis. *Journal of the Royal Statistical Society: Series B (Methodological)*, 20(2):361–372, 2018.
- [64] G.C. Danielson and C. Lanczos. Some improvements in practical fourier analysis and their application to x-ray scattering from liquids. *Journal of the Franklin Institute*, 233(4):365–380, 1942.
- [65] J.W. Cooley, P.A.W. Lewis, and P.D. Welch. Historical notes on the fast fourier transform. *Proceedings of the IEEE*, 55(10):1675–1677, 1967.
- [66] Radix-2-cooley-tukey, 2023. Available from: <https://brianmcfree.net/dstbook-site/content/ch08-fft/FFT.html>.
- [67] NVIDIA Corporation. cufft. CUDA Toolkit, 2023. Available from: <https://developer.nvidia.com/cufft>.
- [68] James B. Birdsong and Nicholas I. Rummelt. The hexagonal fast fourier transform. In *Proceedings of the 2016 IEEE International Conference on Image Processing (ICIP)*, pages 1809–1812, 2016.
- [69] Shlomo Engelberg. Elementary number theory and rader’s fft. *SIAM Review*, 59:671–678, 2017.
- [70] Leo I. Bluestein. A linear filtering approach to the computation of the discrete Fourier transform. *IEEE Northeast Electronics Research and Engineering Meeting*, 10:218–219, 1968.
- [71] Edward H Adelson, Charles H Anderson, James R Bergen, Peter J Burt, and Joan M Ogden. Pyramid methods in image processing. *RCA engineer*, 29(6):33–41, 1984.
- [72] Peter J Burt and Edward H Adelson. The laplacian pyramid as a compact image code. In *Readings in computer vision*, pages 671–679. Elsevier, 1987.
- [73] Aude Oliva, Antonio Torralba, and Philippe G Schyns. Hybrid images. *ACM Transactions on Graphics (TOG)*, 25(3):527–532, 2006.
- [74] Patrick J. Grother. Nist special database 19 - handprinted forms and characters database. Technical report, National Institute of Standards and Technology, 1995. Accessed: 2023-05-31.
- [75] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

- [76] Oren Rippel, Jasper Snoek, and Ryan P Adams. Spectral representations for convolutional neural networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.