



UNIVERSIDAD DE GRANADA

Escuela Técnica Superior de Ingenierías Informática y de
Telecomunicación

Facultad de Ciencias

DOBLE GRADO EN INGENIERÍA INFORMÁTICA Y MATEMÁTICAS

TRABAJO DE FIN DE GRADO

Análisis de Redes Convolucionales y Técnicas de Explicabilidad para Arquitecturas Neuronales Profundas de Segmentación de Instancias

Presentado por:

Juan Manuel Rodríguez Gómez

Tutores:

Pablo Mesejo Santiago

Departamento de Ciencias de la Computación e Inteligencia Artificial

Francisco Javier Merí de la Maza

Departamento de Análisis Matemático

Mentora:

Marilyn Bello García

Departamento de Ciencias de la Computación e Inteligencia Artificial

Curso académico 2024-2025

Análisis de Redes Convolucionales y Técnicas de Explicabilidad para Arquitecturas Neuronales Profundas de Segmentación de Instancias

Juan Manuel Rodríguez Gómez

Juan Manuel Rodríguez Gómez. *Análisis de Redes Convolucionales y Técnicas de Explicabilidad para Arquitecturas Neuronales Profundas de Segmentación de Instancias.*

Trabajo de Fin de Grado. Curso académico 2024-2025.

Responsables de tutorización

Pablo Mesejo Santiago
Departamento de Ciencias de la Computación e Inteligencia Artificial

Doble Grado en Ingeniería Informática y Matemáticas

Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Responsable de mentorización

Francisco Javier Merí de la Maza
Departamento de Análisis Matemático

Facultad de Ciencias

Universidad de Granada

Marilyn Bello García
Departamento de Ciencias de la Computación e Inteligencia Artificial

DECLARACIÓN DE ORIGINALIDAD

D. Juan Manuel Rodríguez Gómez,

Declaro explícitamente que el trabajo presentado como Trabajo de Fin de Grado (TFG), correspondiente al curso académico 2024-2025, es original, entendido esto en el sentido de que no he utilizado para la elaboración del trabajo fuentes sin citarlas debidamente.

En Granada a 25 de noviembre de 2024.

Fdo: Juan Manuel Rodríguez Gómez

Dedicado a mi querida familia, que ha sido mi constante fuente de apoyo y paciencia, a mi pareja, por su amor y comprensión incondicionales, y a mi leal compañero de cuatro patas, cuya presencia y fidelidad han sido una constante alegría y consuelo en los momentos más estresantes de este viaje. Este logro no es solo mío, sino también vuestro.

Agradecimientos

Este trabajo es el fruto de un viaje de constante aprendizaje y desafío, y como tal, hay muchas personas a las que me gustaría expresar mi más sincero agradecimiento.

En primer lugar, agradezco a mis tutores, Francisco Javier Meri de la Maza y Pablo Mesejo Santiago, así como a mi mentora, Marilyn Bello García, cuya guía experta y apoyo han sido esenciales desde la concepción hasta la realización de este proyecto. Sus valiosos consejos, pautas y explicaciones han sido fundamentales para la correcta evolución de este trabajo.

A mis padres, Ana María y Juan Manuel, y a mi hermana y su pareja, Ana Belén y Carlos, por su amor y apoyo incondicional. Su confianza incansable en mí y su capacidad para hacerme ver el lado positivo en cada situación han sido mi refugio y fortaleza durante todo el proceso de esta carrera y, en particular, en la ejecución de este proyecto.

A mi pareja, Nerea, por estar siempre a mi lado, ofreciéndome su infinita paciencia, comprensión y ánimo en los momentos más difíciles de este camino. No hay mayor fortuna que recorrer este camino contigo, y cada logro es aún más valioso porque lo comparto a tu lado.

A mis compañeros de clase, que han compartido conmigo las alegrías y las dificultades de esta etapa académica. Quiero mencionar especialmente a Álex, Álvaro, Higinio, Jesús, Manu y Mónica. Juntos, hemos superado obstáculos y compartido conocimientos, haciendo que cada desafío fuera más llevadero.

A mis amigos, quienes, con su compañía y apoyo constante, han sido una fuente de ánimo y desconexión en los momentos en los que más lo he necesitado. Su presencia ha sido un pilar fundamental en este proceso.

Y finalmente, a Iron, mi fiel compañero de cuatro patas, cuya presencia y compañía leal han sido un dulce consuelo en los largos días de estudio y trabajo.

A todos vosotros, os dedico este logro como una pequeña muestra de mi gran aprecio.

Resumen

PALABRAS CLAVE: Modelización Matemática, Invarianza a Traslaciones, Invarianza frente a Deformaciones, Redes de Dispersión, Teoría de Marcos, Transformada de Ondículas, Ondículas, Aprendizaje Automático, Visión por Computador, Aprendizaje Profundo, Redes Neuronales Convolucionales, Extracción de Características, Segmentación de Instancias, Mask R-CNN, IA Explicable, Grad-CAM.

Por un lado, el propósito matemático principal del trabajo detallado a continuación es desarrollar una modelización matemática general de las redes neuronales convolucionales, de manera que se adapte muchas de las arquitecturas más comúnmente utilizadas para este tipo de redes. Además, utilizando esta modelización general, se demostrará una de las propiedades principales de las redes neuronales convolucionales enfocadas en la extracción de características, la invarianza vertical a traslaciones.

Las redes neuronales convolucionales han llevado a resultados innovadores en numerosas tareas prácticas de Aprendizaje Automático. Muchas de estas aplicaciones realizan primero la extracción de características y luego introducen los resultados en un clasificador entrenable. Su buen rendimiento en este tipo de tareas puede comprobarse empíricamente. Sin embargo, su modelización matemática y la justificación teórica de por qué son buenos para estas tareas sigue siendo un estudio de caso abierto.

El análisis matemático de las redes neuronales convolucionales para la extracción de características fue iniciado por Mallat en su trabajo *Group Invariant Scattering* en 2012 [Mal12]. Específicamente, Mallat consideró las llamadas redes de dispersión basadas en una transformada de ondículas seguida del módulo como función de activación no-lineal en cada capa de la red, y demostró la invarianza a traslaciones (asintóticamente en el parámetro de escala de ondículas) y la invarianza frente a pequeñas deformaciones del extracto de características correspondiente.

La parte matemática de este trabajo se basa en el artículo *A Mathematical Theory of Deep Convolutional Neural Networks for Feature Extraction* presentado por Thomas Wiatowski y Helmut Bölcskei en 2017 [WB18]. Este generaliza los resultados de Mallat desarrollando una teoría que abarca transformaciones convolucionales generales, o en términos más técnicos, marcos semi-discretos generales (incluyendo ondículas, curvículas, crestículas, filtros aprendidos, etc.), funciones de activación no-lineales Lipschitz-continuas generales (por ejemplo, unidades lineales rectificadas, sigmoides logísticos desplazados, tangentes hiperbólicas y funciones de módulo) y operadores de pooling Lipschitz-continuos generales que emulan, por ejemplo, el submuestreo y el pooling promedio. Además, todos estos elementos pueden ser diferentes en las distintas capas de la red. Para la modelización matemática del extracto de características resultante, se demuestra un resultado de invarianza a la traslación de naturaleza vertical en el sentido de que las características se vuelven progresivamente más invariantes a la traslación con el aumento de la profundidad de la red.

Resumen

Por otro lado, la parte informática de este trabajo se centra en la segmentación de instancias, un proceso clave para detectar objetos en una escena y generar máscaras que permitan extraer con precisión dichos objetos. Este proceso implica dos etapas: primero, la detección de la región rectangular que contiene el objeto y, posteriormente, la generación de una máscara que segmenta el objeto detectado, definiendo con precisión sus contornos. En este contexto, la arquitectura Mask R-CNN ha demostrado ser particularmente efectiva para esta tarea, proporcionando resultados prometedores en la segmentación de instancias [HGDG17]. Sin embargo, como muchas redes neuronales profundas, Mask R-CNN presenta limitaciones en términos de interpretabilidad, es decir, la dificultad para comprender las razones detrás de sus decisiones.

Con el objetivo de mejorar la interpretabilidad de Mask R-CNN, este trabajo propone la aplicación de Grad-CAM [SCD⁺17], un método de interpretabilidad post-hoc que permite visualizar las regiones de una imagen que influyen más en las decisiones de la red. La implementación de Grad-CAM sobre Mask R-CNN, basada en adaptaciones propuestas en investigaciones previas [XJ21], [XVM⁺21], y [XVM⁺22], busca facilitar la comprensión de los procesos de inferencia de esta arquitectura neuronal. De este modo, en este trabajo se pretende dotar de explicabilidad a un modelo que de otro modo sería una caja negra.

Summary

KEYWORDS: Mathematical Modeling, Translation Invariance, Deformation Invariance, Scattering Networks, Frame Theory, Wavelet Transform, Wavelets, Machine Learning, Computer Vision, Deep Learning, Convolutional Neural Networks, Feature Extraction, Instance Segmentation, Mask R-CNN, Explainable AI, Grad-CAM.

On one hand, the main mathematical purpose of the work detailed below is to develop a comprehensive mathematical modeling of convolutional neural networks, adapting to many of the most commonly used architectures for this type of network. Additionally, using this general modeling, one of the main properties of convolutional neural networks focused on feature extraction, namely translation invariance, will be demonstrated.

Convolutional neural networks have led to groundbreaking results in numerous practical machine learning tasks. Many of these applications first perform feature extraction and then feed the results into a trainable classifier. Their good performance in such tasks can be empirically verified. However, their mathematical modeling and theoretical justification for their effectiveness in these tasks remain an open area of research.

The mathematical analysis of convolutional neural networks for feature extraction was initiated by Mallat in his work *Group Invariant Scattering* in 2012 [Mal12]. Specifically, Mallat considered the so-called scattering networks based on a wavelet transform followed by the modulus as a non-linear activation function in each layer of the network, and demonstrated translation invariance (asymptotically in the wavelet scale parameter) and invariance against small deformations of the corresponding feature extractor.

The mathematical part of this work builds on the article *A Mathematical Theory of Deep Convolutional Neural Networks for Feature Extraction* by Thomas Wiatowski and Helmut Bölcskei, presented in 2017 [WB18]. This work generalizes Mallat's results by developing a theory that encompasses general convolutional transformations, or in more technical terms, general semi-discrete frames (including wavelets, curvelets, ridgelets, learned filters, etc.), general Lipschitz-continuous non-linear activation functions (e.g., rectified linear units, shifted logistic sigmoids, hyperbolic tangents, and modulus functions), and general Lipschitz-continuous pooling operators that emulate, for example, subsampling and averaging. Moreover, all these elements can differ across the network's layers. For the mathematical modeling of the resulting feature extractor, a vertical translation invariance result is demonstrated, meaning that the features become progressively more invariant to translation as the network depth increases.

On the other hand, the computational part of this work focuses on instance segmentation, a key process for detecting objects in a scene and generating masks that precisely extract these objects. This process involves two stages: first, detecting the rectangular region containing the object, and then generating a mask that segments the detected object, accurately defining

Summary

its contours. In this context, the Mask R-CNN architecture has proven to be particularly effective for this task, providing promising results in instance segmentation [HGDG17]. However, like many deep neural networks, Mask R-CNN faces challenges in terms of interpretability, meaning the difficulty in understanding the reasoning behind its decisions.

To improve the interpretability of Mask R-CNN, this work proposes the application of Grad-CAM [SCD⁺17], a post-hoc interpretability method that allows for visualizing the regions of an image that most influence the network's decisions. The implementation of Grad-CAM on Mask R-CNN, based on adaptations proposed in previous research [XJ21], [XVM⁺21], and [XVM⁺22], aims to enhance the understanding of the inference processes of this neural architecture. Thus, this work intends to provide explainability to a model that would otherwise be a black box.

Índice general

Agradecimientos	v
Resumen	VII
Summary	IX
Índice de figuras	XV
Índice de tablas	XIX
I. Análisis de Redes Convolucionales	1
1. Introducción	3
2. Teoría de Mallat sobre Redes de Dispersión	9
2.1. Evolución matemática: De Fourier a Littlewood-Paley	9
2.1.1. La transformada de Fourier y sus limitaciones	9
2.1.2. Ondículas como alternativa a la transformada de Fourier	12
2.1.3. La transformada de Littlewood-Paley	15
2.1.4. Convenciones para futuras secciones	20
2.2. El operador de dispersión sobre un camino ordenado	21
2.2.1. Obtención de coeficientes invariantes por traslaciones	21
2.2.2. El operador módulo	24
2.2.3. El propagador de dispersión	25
2.2.4. La transformada de dispersión	26
2.3. Propagación de la dispersión y conservación de la norma	28
2.3.1. Proceso iterativo del propagador de la dispersión	28
2.3.2. Comparativa con una red neuronal convolucional	30
2.3.3. No-expansividad en la distancia entre funciones	30
2.3.4. Conservación de la norma	32
2.4. Invarianza horizontal por traslaciones	37
2.4.1. No-expansividad en conjuntos de caminos	37
2.4.2. Demostración de la invarianza por traslaciones	40
2.5. Invarianza frente a pequeñas deformaciones	45
2.5.1. Cotas superiores en comutadores de dispersión	45
2.5.2. Demostración de la invarianza frente a pequeñas deformaciones	47
3. Generalización de la Teoría de Mallat sobre Redes de Dispersión	51
3.1. Resumen de la teoría de Mallat sobre redes de dispersión	51
3.1.1. Definición del vector de características en redes de dispersión	51
3.1.2. Marcos semi-discretos	52

Índice general

3.1.3. Relación entre la arquitectura de redes de dispersión y las características extraídas	54
3.2. Construcción del extractor de características convolucional general	55
3.2.1. Diferencias y similitudes con las redes de dispersión	55
3.2.2. Operadores de pooling	57
3.3. Invarianza vertical por traslaciones	65
4. Conclusiones y Trabajos Futuros	75
II. Técnicas de Explicabilidad para Arquitecturas Neuronales Profundas de Segmentación de Instancias	77
5. Introducción	79
5.1. Descripción del problema y motivación	79
5.2. Objetivos	80
5.3. Planificación y estimación de costes	81
6. Fundamentos Teóricos	83
6.1. Aprendizaje Automático	83
6.1.1. Aprendizaje Supervisado	84
6.1.2. Aprendizaje No Supervisado	88
6.2. Aprendizaje Profundo	88
6.2.1. Redes neuronales	88
6.2.2. Back-propagation	91
6.3. Redes neuronales convolucionales	92
6.3.1. Capa convolucional	94
6.3.2. Capa de pooling	95
6.3.3. Capa totalmente conectada	96
6.3.4. Batch normalization	98
6.3.5. Proceso de entrenamiento	98
6.3.6. Fine-tuning	98
6.4. Visión por Computador	99
6.4.1. Clasificación de imágenes	99
6.4.2. Detección de objetos	101
6.4.3. Segmentación semántica	103
6.4.4. Segmentación de instancias	105
6.5. Arquitecturas R-CNN	108
6.5.1. Arquitectura R-CNN	108
6.5.2. Arquitectura Fast R-CNN	109
6.5.3. Arquitectura Faster R-CNN	111
6.6. Inteligencia Artificial Explicable	112
6.6.1. Motivación y conceptos fundamentales	112
6.6.2. Tipos de explicabilidad	113
6.6.3. Propiedades deseables	114
6.6.4. Técnicas de explicabilidad post-hoc en redes neuronales convolucionales	114
7. Estado del Arte	117

8. Métodos Propuestos	121
8.1. Mask R-CNN	121
8.1.1. Backbone: Red de extracción de características	121
8.1.2. Red de propuestas de regiones	125
8.1.3. Capa de alineación de regiones de interés	127
8.1.4. Network head: Ramas de predicciones y segmentación	128
8.1.5. Funciones de pérdida	129
8.2. Grad-CAM	131
8.2.1. Funcionamiento	132
8.2.2. Ventajas y limitaciones	133
9. Implementación	135
9.1. Diseño del software	135
9.2. Entorno de ejecución	135
10. Experimentación	137
10.1. Datos empleados	137
10.2. Protocolo de validación experimental	137
10.3. Métricas empleadas	137
10.4. Experimentos realizados	137
10.5. Resultados obtenidos	137
10.6. Discusión de los resultados obtenidos	137
11. Conclusiones y Trabajos Futuros	139
Bibliografía	141

Índice de figuras

1.1.	Dígitos manuscritos del conjunto de datos MNIST [LC98]. Ejemplo de invarianza por traslaciones. A pesar de la variación en la ubicación del dígito manuscrito 5 en las imágenes de la izquierda y la derecha, este se identifica de manera consistente. Imagen extraída de [KA18].	3
1.2.	Dígitos manuscritos del conjunto de datos MNIST [LC98]. Ejemplo de invarianza frente a pequeñas deformaciones. A pesar de las variaciones en la forma de los dígitos manuscritos, estos se identifican correctamente. Imagen extraída de [LYP16].	4
1.3.	Dígitos manuscritos del conjunto de datos MNIST [LC98]. Ejemplos de fallos de predicción causados por deformaciones en los dígitos. Las deformaciones pueden llevar a confusiones entre diferentes clases, como se observa en las imágenes donde la predicción (Pred) y la etiqueta real (GT) no coinciden. Por este motivo, nos centramos en pequeñas deformaciones, para no alterar la identidad del objeto en la imagen. Imagen extraída de [ZKS ⁺ 19].	5
2.1.	Representación gráfica de la función base de Haar. Imagen extraída de [Wik24b].	13
2.2.	Filtro generado por el soporte de una ondícula para detectar bordes (1) verticales, (2) horizontales y (3) diagonales. Imagen elaborada por el autor.	14
2.3.	Ejemplo de aplicación de la base de Haar a una imagen (arriba izquierda). Los filtros destacan los bordes en tres direcciones: horizontal (arriba derecha), vertical (abajo izquierda) y diagonal (abajo derecha). Imagen extraída de [Jor24].	14
2.4.	Como se observa en el caso de la izquierda, la ondícula está afectada por una escala más pequeña, lo que le permite detectar cambios en la señal con mayor precisión a frecuencias más altas a lo largo del tiempo mediante la convolución. En contraste, en el caso de la derecha, la ondícula está sujeta a una escala mayor, lo que reduce su capacidad para detectar con precisión las variaciones en la señal. Imagen extraída de [Mat24a].	17
2.5.	Un propagador de dispersión U_J aplicado a un punto de una señal $f(x)$ calcula $(U[\lambda_1]f)(x) = (f * \psi_{\lambda_1})(x) $. En la capa $m = 0$, se promedian los coeficientes que dieron 0 (por tener $2^j < 2^{-J}$), obteniendo como salida $(S_J[\emptyset]f)(x) = (f * \phi_{2^J})(x)$ (ver flecha negra). Luego, se aplica U_J a cada coeficiente $(U[\lambda_1]f)(x)$ del paso anterior, calculando así $(U[\lambda_1, \lambda_2]f)(x)$ y obteniendo como salida $(S_J[\lambda_1]f)(x) = ((U[\lambda_1]f) * \phi_{2^J})(x)$ en la capa $m = 1$. Repitiendo este proceso de forma recursiva para cada coeficiente $(U[p]f)(x)$ se va obteniendo $(S_J[p]f)(x) = ((U[p]f)(x) * \phi_{2^J})$ como salida de cada capa. Imagen extraída de [BM13].	29
3.1.	Partición del plano de frecuencias \mathbb{R}^2 inducida por un marco semi-discreto de ondículas direccionales, con $K = 12$ direcciones. Imagen extraída de [WB18]. .	54
3.2.	Arquitectura de red de dispersión basada en filtros de ondículas y la no-linealidad del módulo. Los elementos del vector de características $\Phi_W(f)$ se indican en las puntas de las flechas. Imagen extraída de [WB18].	55

3.3. Arquitectura de red subyacente al extractor de características convolucional general. El índice $\lambda_n^{(k)}$ corresponde al k -ésimo átomo $g_{\lambda_n^{(k)}}$ del marco Ψ_n asociado a la capa n -ésima de la red. La función χ_n es el átomo generador de salida de la capa n -ésima. Imagen extraída de [WB18].	61
3.4. Dígitos manuscritos del conjunto de datos MNIST [LC98]. Para tareas prácticas de Aprendizaje Automático (por ejemplo, clasificación de señales), a menudo deseamos que el vector de características $\Phi_\Omega(f)$ sea invariante a la ubicación espacial de los dígitos dentro de la imagen f . El Teorema 3.1 establece que las características $\Phi_\Omega^n(f)$ se vuelven más invariantes por traslaciones conforme aumenta el índice de la capa n . Imagen extraída de [WB18].	70
3.5. Dígitos manuscritos del conjunto de datos MNIST [LC98]. Si f representa la imagen del dígito 5 en (a), entonces, con una función τ seleccionada apropiadamente, la función $T_\tau f = f(\cdot - \tau(\cdot))$ modela imágenes del 5 basadas en diferentes estilos de escritura como en (b) y (c). Imagen extraída de [WB18].	73
5.1. La segmentación de instancias combina la detección de objetos y la segmentación semántica. Imagen extraída de [Liu18].	79
5.2. Visualización de mapas de calor generados por Grad-CAM para una imagen de entrada que contiene un perro y un gato. La imagen central muestra el mapa de calor correspondiente a la predicción de la clase <i>Cat</i> , destacando las áreas relevantes para dicha clasificación. La imagen de la derecha presenta el mapa de calor para la clase <i>Dog</i> , resaltando las zonas de la imagen que la red considera importantes para identificar al perro. Imagen extraída de [SCD ⁺ 17].	80
6.1. Importancia de una elección adecuada de la tasa de aprendizaje. Imagen extraída de [Unizo].	87
6.2. Perceptrón multicapa con una capa oculta. Formalmente, se puede expresar como $\hat{f}(x, w) = f_2(f_1(x, w))$, donde f_1 representa la transformación de los datos de la capa de entrada a la capa oculta, y f_2 es la transformación de la capa oculta a la capa de salida. Imagen extraída de [Wik24c].	89
6.3. Gráficas de las funciones de activación más utilizadas. Imagen extraída de [Onl24].	91
6.4. Ejemplo de back-propagation representando la función $f(x, y, z) = (x + y)z$, para valores $x = -2, y = 5, z = -4$, en un grafo computacional. El grafo muestra el flujo de cálculo de la salida $f = -12$. Primero se calculan los valores desde las entradas hasta la salida (mostrados en verde). Luego, se realiza el back-propagation, que comienza al final y aplica recursivamente la regla de la cadena para calcular los gradientes (mostrados en rojo) hasta llegar a las entradas del circuito. Imagen extraída de [Unizo].	92
6.5. Diagrama esquemático de la arquitectura de una red neuronal convolucional, que ilustra el proceso de aprendizaje de características y la etapa de clasificación. Las capas de convolución y pooling permiten la extracción progresiva de características significativas, mientras que las capas finales totalmente conectadas realizan la clasificación en categorías predefinidas, como coche o bicicleta. Imagen extraída de [Mat24b].	94

6.6. Ejemplo visual del flujo de datos en una red neuronal convolucional, donde se observa cómo las capas de convolución (CONV) y activación (ReLU) extraen características de la imagen de entrada, seguidas de capas de pooling (POOL) que reducen la dimensión espacial manteniendo la profundidad. Al final, una capa totalmente conectada (FC) clasifica la imagen en una categoría específica, siendo coche en este caso. Imagen extraída de [Uni20].	95
6.7. Ejemplo que ilustra cómo se calcula un mapa de activación en una capa convolucional para un volumen de entrada específico. Imagen extraída de [Uni20].	96
6.8. Secuencia de varias capas convolucionales seguidas de la función de activación. Es importante señalar que la profundidad de los filtros siempre coincide con la del volumen de entrada. Imagen extraída de [Uni20].	96
6.9. La capa de pooling reduce espacialmente el tamaño del volumen de entrada de forma independiente en cada capa de profundidad. Izquierda: En este ejemplo, el volumen de entrada de tamaño $224 \times 224 \times 64$ se reduce mediante un filtro de tamaño 2 y un stride de 2, obteniendo un volumen de salida de tamaño $112 \times 112 \times 64$. Nótese que la profundidad del volumen se mantiene. Derecha: La operación de submuestreo más común es el max-pooling, que en este caso se realiza con un stride de 2. Es decir, se toma el valor máximo de entre 4 números (un pequeño cuadrado de 2×2). Imagen extraída de [Uni20].	97
6.10. Ejemplo de capa totalmente conectada con dos capas ocultas. Imagen extraída de [Uni20].	97
6.11. Diferentes tareas de Visión por Computador: Clasificación de imágenes, detección de objetos, segmentación semántica y segmentación de instancias aplicadas en una misma escena. Imagen extraída de [LOW ⁺ 19].	100
6.12. Resultado de aplicar clasificación de imágenes en una fotografía que contiene dos categorías: Caballo y Persona. Imagen extraída de [Sor18].	101
6.13. Resultado de aplicar detección de objetos en una fotografía que contiene dos categorías: Caballo y Persona. Las bounding boxes azules delimitan a las personas detectadas, mientras que las amarillas localizan a los caballos en la imagen. Imagen extraída de [Sor18].	102
6.14. Resultado de aplicar segmentación semántica en una fotografía que contiene dos categorías: Caballo y Persona. Todos los píxeles que pertenecen a la misma clase se muestran en una misma máscara, sin diferenciar entre instancias. Imagen extraída de [Sor18].	104
6.15. Resultado de aplicar segmentación de instancias en una fotografía que contiene dos categorías: Caballo y Persona. Todos los píxeles que pertenecen a la misma instancia se muestran en una misma máscara. El resultado final es una máscara por instancia. Imagen extraída de [Sor18].	105
6.16. Arquitectura R-CNN. Imagen extraída de [GDDM13].	109
6.17. Arquitectura Fast R-CNN. Imagen extraída de [Gir15].	110
6.18. Arquitectura Faster R-CNN. Imagen extraída de [DSZ ⁺ 18].	111
6.19. Precisión frente a interpretabilidad para distintos algoritmos de Aprendizaje Automático. Imagen extraída de [ASJ ⁺ 21].	112

Índice de figuras

7.1. Número de publicaciones por año obtenidas en Scopus con la búsqueda realiza el 3 de noviembre de 2024, utilizando términos de búsqueda relacionados con segmentación de instancias y XAI. Se observa un aumento significativo en las publicaciones a partir de 2019, reflejando el creciente interés en la expli-cabilidad aplicada a arquitecturas neuronales profundas de segmentación de instancias. Imagen extraída de [Sco24].	118
8.1. Arquitectura Mask R-CNN. Imagen extraída de [SKY23].	122
8.2. Bloque residual de dos capas. Imagen extraída de [HZRS15].	124
8.3. Bloque residual de tres capas (también conocido como estructura bottleneck). Imagen extraída de [HZRS15].	125
8.4. El backbone que utiliza la estructura FPN está formado por rutas ascendentes y descendentes conectadas mediante conexiones de salto. Imagen extraída de [LDG ⁺ 16].	126
8.5. Ejemplo de interpolación por vecino más cercano. Imagen elaborada por el autor.	126
8.6. Las distintas anchor boxes generadas tienen diferentes escalas y relaciones de aspecto. Imagen extraída de [Bup18].	128
8.7. Comparación entre RoI Pooling (izquierda) y RoI Align (derecha). Imagen extraída de [Hui18a].	129
8.8. Resultados de Mask R-CNN en el conjunto de test de COCO. Las máscaras se muestran en diferentes colores, e incluyen también la bounding box, la categoría y las puntuaciones de confianza. Imagen extraída de [HGDG17]. . .	130
8.9. Grad-CAM aplicado en diferentes capas convolucionales para la clase <i>tiger cat</i> . Esta figura examina cómo las localizaciones varían cualitativamente al aplicar Grad-CAM en distintos mapas de características de una red neuronal convolucional (VGG16 [SZ14]). Observamos que las visualizaciones de mejor calidad se obtienen tras la capa convolucional más profunda de la red, mientras que la precisión de las localizaciones disminuye en capas más su-perficiales. Esto concuerda con lo comentado anteriormente, según lo cual las capas convolucionales más profundas capturan conceptos semánticos más complejos. Imagen extraída de [SCD ⁺ 17].	133

Índice de tablas

Parte I.

Análisis de Redes Convolucionales

1. Introducción

Las redes neuronales convolucionales se han establecido como una herramienta preeminente en el ámbito de la Inteligencia Artificial, específicamente en el campo del Aprendizaje Profundo. Esta rama del Aprendizaje Automático ha experimentado un crecimiento sustancial en investigación y publicaciones, especialmente debido a la eficacia notable de las redes neuronales convolucionales en el Procesamiento de Imágenes para tareas tales como clasificación, segmentación y generación de imágenes. Por ello, en este trabajo se busca profundizar en la modelización matemática de estas redes para entender mejor sus propiedades teóricas y validar una de sus capacidades más cruciales: la invarianza por traslaciones.

En el contexto del Aprendizaje Automático y la Visión por Computador, un desafío fundamental es la clasificación y detección de objetos en imágenes. Definimos así la invarianza como la habilidad de identificar un objeto en una imagen a pesar de cambios en su apariencia, como rotaciones, deformaciones leves o traslaciones. Esta característica es vital ya que asegura que la identidad del objeto se mantiene constante a pesar de estas variaciones.

Específicamente, la invarianza por traslaciones se describe como la capacidad de reconocer un objeto en cualquier lugar de la imagen (véase Figura 1.1). Esta propiedad es esencial y sabemos empíricamente que las redes neuronales convolucionales la cumplen.

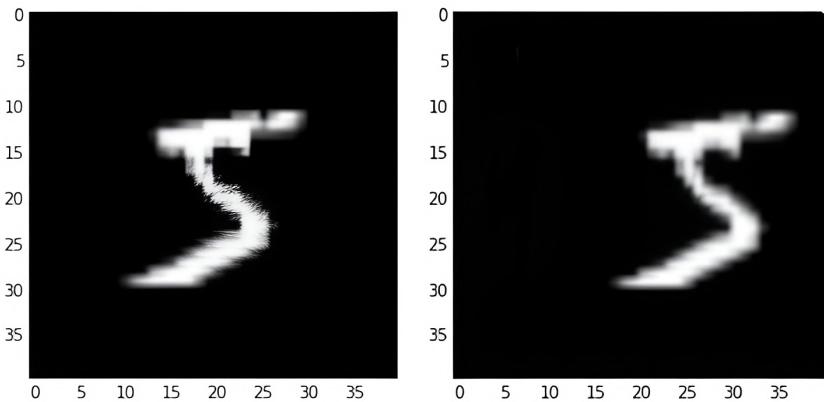


Figura 1.1.: Dígitos manuscritos del conjunto de datos MNIST [LC98]. Ejemplo de invarianza por traslaciones. A pesar de la variación en la ubicación del dígito manuscrito 5 en las imágenes de la izquierda y la derecha, este se identifica de manera consistente. Imagen extraída de [KA18].

Definición 1.1. Sea $(T_t f)(x) := f(x - t)$ la traslación de $f \in L^2(\mathbb{R}^d)$ por $t \in \mathbb{R}^d$, donde

1. Introducción

$L^2(\mathbb{R}^d) := \left\{ f : \mathbb{R}^d \rightarrow \mathbb{C} \mid \int_{\mathbb{R}^d} |f(x)|^2 dx < \infty \right\}$. Sea \mathcal{H} un espacio de Hilbert (espacio vectorial completo con producto escalar que induce una norma). Decimos que un operador $\Phi : L^2(\mathbb{R}^d) \rightarrow \mathcal{H}$ es *invariante por traslaciones* si $\Phi(T_t f) = \Phi(f)$ para todo $f \in L^2(\mathbb{R}^d)$ y para todo $t \in \mathbb{R}^d$.

Adicionalmente, la invarianza frente a pequeñas deformaciones (difeomorfismos) también es una propiedad cumplida por las redes neuronales convolucionales, permitiendo que un objeto sea identificado correctamente a pesar de cambios menores en su forma (véase [Figura 1.2](#) y [Figura 1.3](#)).

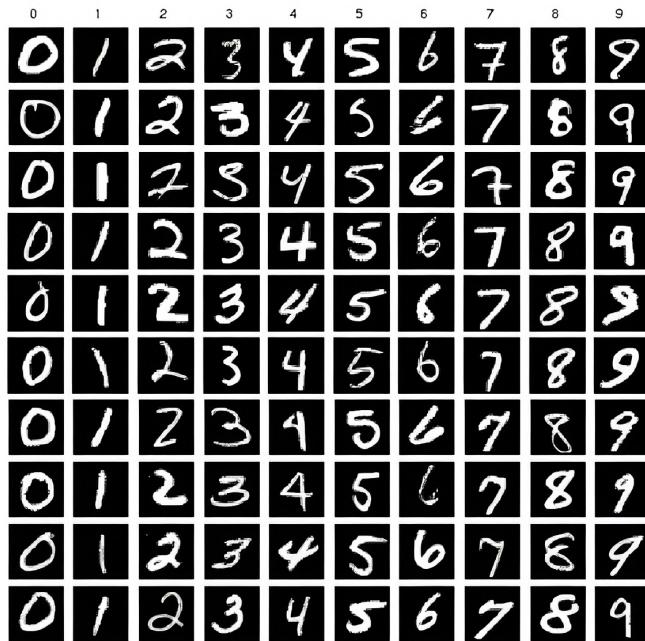


Figura 1.2.: Dígitos manuscritos del conjunto de datos MNIST [LC98]. Ejemplo de invarianza frente a pequeñas deformaciones. A pesar de las variaciones en la forma de los dígitos manuscritos, estos se identifican correctamente. Imagen extraída de [LYP16].

Definición 1.2. Decimos que una función diferenciable $f : \mathbb{R}^d \rightarrow \mathbb{R}^d$ es un *difeomorfismo* si f es una biyección y su inversa $f^{-1} : \mathbb{R}^d \rightarrow \mathbb{R}^d$ también es diferenciable.

En el siguiente apartado se analizará el caso del módulo de la transformada de Fourier de una función f , empleándolo como un ejemplo de un operador que mantiene la invarianza por traslaciones. Sin embargo, se observará que las inestabilidades que surgen frente a deformaciones en las frecuencias altas nos llevan a descartarlo como una opción viable para la modelización de las redes neuronales convolucionales, ya que no garantiza la Lipschitz-continuidad ante difeomorfismos. Es bien sabido que las inestabilidades a las deformaciones

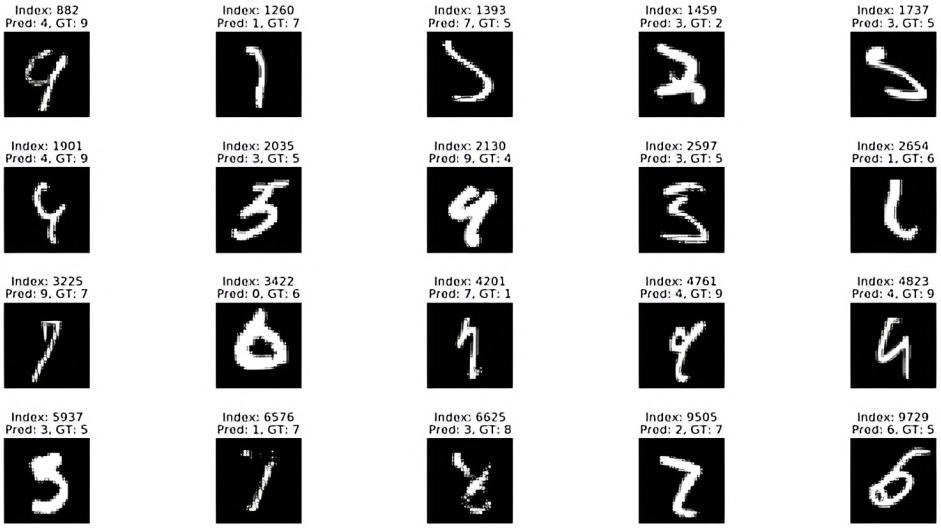


Figura 1.3.: Dígitos manuscritos del conjunto de datos MNIST [LC98]. Ejemplos de fallos de predicción causados por deformaciones en los dígitos. Las deformaciones pueden llevar a confusiones entre diferentes clases, como se observa en las imágenes donde la predicción (Pred) y la etiqueta real (GT) no coinciden. Por este motivo, nos centramos en pequeñas deformaciones, para no alterar la identidad del objeto en la imagen. Imagen extraída de [ZKS⁺19].

aparecen a altas frecuencias [Hö71]. La mayor dificultad es mantener la Lipschitz-continuidad en dichas altas frecuencias.

Para preservar la estabilidad en $L^2(\mathbb{R}^d)$ queremos que Φ sea no-expansivo.

Definición 1.3. Sea \mathcal{H} un espacio de Hilbert. Decimos que un operador $\Phi : L^2(\mathbb{R}^d) \rightarrow \mathcal{H}$ es *no-expansivo* si:

$$\forall f, h \in L^2(\mathbb{R}^d), \quad \|\Phi(f) - \Phi(h)\| \leq \|f - h\|,$$

donde $\|f\| := (\int |f(x)|^2 dx)^{1/2}$ denota la norma de f en $L^2(\mathbb{R}^d)$.

Este estudio se centrará en funciones $L^2(\mathbb{R}^d)$, donde se buscará desarrollar un operador que sea Lipschitz-continuo bajo la influencia de difeomorfismos y que conserve información esencial de alta frecuencia para distinguir entre distintos tipos de señales. Dentro de este marco, vamos a encargarnos de verificar la Lipschitz-continuidad relativa a la acción de pequeños difeomorfismos cercanos a las traslaciones. Dichos difeomorfismos transformarán un punto $x \in \mathbb{R}^d$ en $x - \tau(x)$, donde τ es el campo de desplazamiento. Denotaremos $(T_\tau f)(x) := f(x - \tau(x))$ como la acción del difeomorfismo $\mathbb{1} - \tau$ sobre f .

Es importante recordar que la condición de Lipschitz se define como:

Definición 1.4. Sea $f : M \rightarrow N$ una función entre dos espacios métricos M y N con sus respectivas distancias d_M y d_N . Decimos que f satisface la condición de Lipschitz si:

$$\exists C > 0 \quad \text{tal que} \quad d_N(f(x), f(y)) \leq Cd_M(x, y), \quad \forall x, y \in M.$$

En nuestro caso particular, dado que el espacio de partida es $L^2(\mathbb{R}^d)$ y las funciones que vamos a comparar son f y $T_\tau f$, sabemos que $\|\Phi(f) - \Phi(T_\tau f)\|$ estará limitada por $\|f\| d(\mathbb{1}, \mathbb{1} - \tau)$. Por lo tanto, necesitamos establecer una medida de distancia entre $\mathbb{1}$ y $\mathbb{1} - \tau$.

Definición 1.5. Se define una *distancia entre $\mathbb{1} - \tau$ y $\mathbb{1}$* en cualquier subconjunto compacto Ω de \mathbb{R}^d como:

$$d_\Omega(\mathbb{1}, \mathbb{1} - \tau) = \sup_{x \in \Omega} |\tau(x)| + \sup_{x \in \Omega} |(Jac(\tau))(x)| + \sup_{x \in \Omega} |(Hess(\tau))(x)|,$$

donde $|\tau(x)|$ denota la norma euclídea en \mathbb{R}^d , $|(Jac(\tau))(x)|$ la norma del supremo de la matriz Jacobiana, y $|(Hess(\tau))(x)|$ la norma del supremo del tensor Hessiano.

Dado que trabajaremos con funciones que son invariantes por traslaciones, la condición de Lipschitz no dependerá de la magnitud máxima de la traslación $\|\tau\|_\infty := \sup_{x \in \mathbb{R}^d} |\tau(x)|$. Por esta razón, omitiremos este término. De este modo, podemos formular finalmente la condición de Lipschitz que un operador debe cumplir en nuestro caso:

Definición 1.6. Sea \mathcal{H} un espacio de Hilbert. Un operador invariante por traslaciones $\Phi : L^2(\mathbb{R}^d) \rightarrow \mathcal{H}$ se dice *Lipschitz-continuo* por la acción de los difeomorfismos C^2 si para cualquier compacto $\Omega \subset \mathbb{R}^d$ existe una constante $c > 0$ tal que para todo $f \in L^2(\mathbb{R}^d)$ con soporte en Ω y para todo $\tau \in C^2(\mathbb{R}^d)$ se cumple:

$$\|\Phi(f) - \Phi(T_\tau f)\| \leq c \|f\| (\|Jac(\tau)\|_\infty + \|Hess(\tau)\|_\infty), \quad (1.1)$$

donde $\|Jac(\tau)\|_\infty := \sup_{x \in \mathbb{R}^d} |(Jac(\tau))(x)|$ y $\|Hess(\tau)\|_\infty := \sup_{x \in \mathbb{R}^d} |(Hess(\tau))(x)|$.

La continuidad Lipschitz de (1.1) implica que Φ es invariante por traslaciones globales, pero dicha condición es mucho más fuerte: (1.1) garantiza que Φ se ve poco afectada por los términos de primer y segundo grado de difeomorfismos que son traslaciones locales.

Tras haber introducido las principales herramientas con las que trabajaremos, veremos en las próximas secciones cómo la solución al problema implicará el uso de transformadas de ondículas. No obstante, esto introduce nuevos desafíos, como el hecho de que estas transformadas no son invariantes bajo traslaciones. Para lograr dicha invarianza, será necesario combinar la transformada con un operador no-lineal que nos permita obtener coeficientes invariantes. Este operador se construirá mediante una serie de convoluciones con operadores no-lineales y no comutativos, donde cada uno calculará el módulo de la transformada de ondículas. Este nuevo operador podrá interpretarse como la modelización matemática de una red neuronal

convolucional, pero no de cualquiera en general, sino un tipo particular, con el operador módulo como función de activación tras cada capa convolucional y sin ninguna capa de pooling.

Una vez realizada la modelización anterior, el objetivo de este trabajo será generalizar la teoría de Mallat sobre redes de dispersión considerando redes convolucionales más generales, con diferentes tipos de filtros convolucionales, funciones de activación no-lineales y operadores de pooling. Finalmente, estudiaremos su invarianza vertical por translaciones, en el sentido de que las características se vuelven progresivamente más invariantes a las translaciones a medida que aumenta la profundidad de la red.

2. Teoría de Mallat sobre Redes de Dispersión

Nuestro primer objetivo será abordar la formulación matemática de una red neuronal convolucional con el operador módulo como función de activación tras cada capa convolucional y sin ninguna capa de pooling. Para ello, necesitamos definir un operador, al que llamaremos propagador de dispersión, que se aplicará de manera recursiva en una cascada de convoluciones. También discutiremos los desafíos asociados con la elección de un operador que sea continuo de acuerdo con la condición de Lipschitz bajo la acción de difeomorfismos y que sea invariante bajo traslaciones. Esto ayudará a evitar problemas como la inestabilidad de altas frecuencias que puede surgir en las señales afectadas por difeomorfismos.

En segundo lugar, exploraremos posibles soluciones para prevenir la aparición de estas inestabilidades, utilizando bases derivadas de la transformada de ondículas de Littlewood-Paley. Con esta opción, conseguiremos un operador Lipschitz-continuo bajo la acción de difeomorfismos.

A continuación, nuestro objetivo será obtener coeficientes que permanezcan invariantes ante traslaciones. Para lograrlo, será necesario emplear un operador no-lineal, como es el módulo.

Una vez hayamos definido un operador que cumpla con todas las propiedades mencionadas, presentaremos el propagador de dispersión. Este será el resultado de aplicar dicho operador en cadena a lo largo de un “camino” de frecuencias y rotaciones, lo que nos permitirá describir matemáticamente una red neuronal convolucional con el operador módulo como función de activación tras cada capa convolucional y sin ninguna capa de pooling.

Este capítulo se basa en la investigación de Mallat, siguiendo como referencia su trabajo, [Mal12] y [Malo8], junto con las contribuciones de otros autores que serán citados de forma adecuada.

2.1. Evolución matemática: De Fourier a Littlewood-Paley

2.1.1. La transformada de Fourier y sus limitaciones

El Análisis de Fourier ha sido históricamente esencial en el campo del Procesamiento de Señales, lo que sugiere que la transformada de Fourier podría ser un excelente punto de inicio para desarrollar un propagador de dispersión, dado su reconocido poder en este ámbito. La idea principal detrás de la transformada de Fourier es representar funciones no periódicas, que sin embargo tienen un área definida bajo su curva, mediante la suma de senos y cosenos, cada uno ponderado por una función específica que asigna su importancia en cada momento.

Definición 2.1. Sea $f \in L^1(\mathbb{R}^d)$, donde $L^1(\mathbb{R}^d) := \left\{ f : \mathbb{R}^d \rightarrow \mathbb{C} \mid \int_{\mathbb{R}^d} |f(x)| dx < \infty \right\}$. Se define la *transformada de Fourier* de f como la función:

2. Teoría de Mallat sobre Redes de Dispersión

$$\widehat{f}(\omega) := \int_{\mathbb{R}^d} f(x) e^{-2\pi i \langle x, \omega \rangle} dx = \int_{\mathbb{R}^d} f(x) [\cos(2\pi \langle x, \omega \rangle) - i \sin(2\pi \langle x, \omega \rangle)] dx,$$

donde $\langle x, y \rangle$ denota el producto escalar entre $x, y \in \mathbb{R}^d$.

La **Definición 2.1** se extiende en el sentido usual a $L^2(\mathbb{R}^d)$ gracias al *teorema de Plancherel* ([Rud91], Teorema 7.9), que establece que la transformada de Fourier actúa como una isometría lineal en $L^2(\mathbb{R}^d)$, preservando la norma cuadrática de las funciones. Esta extensión se basa en que las funciones en $L^1(\mathbb{R}^d) \cap L^2(\mathbb{R}^d)$ son densas en $L^2(\mathbb{R}^d)$, lo que garantiza la continuidad y coherencia de la transformada al pasar de $L^1(\mathbb{R}^d)$ a $L^2(\mathbb{R}^d)$.

Una de las características más notables de la transformada de Fourier es su capacidad para reconstruir una función a partir de su representación en el dominio de las frecuencias sin ninguna pérdida de datos. Esto hace posible operar en el denominado dominio de Fourier, también llamado dominio de frecuencia, donde la función transformada depende únicamente de la variable de frecuencia ω . Posteriormente, es factible retornar al dominio original de la función utilizando la transformada inversa de Fourier, también sin incurrir en pérdidas de información.

Esta característica inicialmente resulta bastante ventajosa, ya que permite manejar la información en un dominio más simple y deducir resultados que se pueden revertir al dominio original de la señal sin comprometer la integridad de la información. Además, en el análisis de señales, es común utilizar el módulo de la transformada de Fourier para simplificar el análisis al evitar las fases complejas. Por lo tanto, el primer operador que vamos a evaluar es el módulo de la transformada de Fourier.

Definición 2.2. Sea \mathcal{H} un espacio de Hilbert. Definimos el operador $\Phi : L^2(\mathbb{R}^d) \rightarrow \mathcal{H}$, $\Phi(f) := |\widehat{f}|$, como el *módulo de la transformada de Fourier*.

Primero, para determinar si este operador es adecuado para nuestros objetivos, necesitamos confirmar que sea invariante por traslaciones.

Lema 2.1. *El operador $\Phi(f) = |\widehat{f}|$ es invariante por traslaciones.*

Demostración. Para cada $t \in \mathbb{R}^d$, consideramos la traslación $(T_t f)(x) = f(x - t)$. Se tiene que:

$$\widehat{(T_t f)}(\omega) = \int_{\mathbb{R}^d} (T_t f)(x) e^{-2\pi i \langle x, \omega \rangle} dx = \int_{\mathbb{R}^d} f(x - t) e^{-2\pi i \langle x, \omega \rangle} dx.$$

Realizando el cambio de variable $y = x - t$, se obtiene:

$$\int_{\mathbb{R}^d} f(x - t) e^{-2\pi i \langle x, \omega \rangle} dx = \int_{\mathbb{R}^d} f(y) e^{-2\pi i \langle (y + t), \omega \rangle} dy$$

$$\begin{aligned}
 &= \int_{\mathbb{R}^d} f(y) e^{-2\pi i \langle y, \omega \rangle} e^{-2\pi i \langle t, \omega \rangle} dy \\
 &= e^{-2\pi i \langle t, \omega \rangle} \int_{\mathbb{R}^d} f(y) e^{-2\pi i \langle y, \omega \rangle} dy \\
 &= e^{-2\pi i \langle t, \omega \rangle} \widehat{f}(\omega).
 \end{aligned}$$

Por tanto, $|\widehat{(T_t f)}(\omega)| = |e^{-2\pi i \langle t, \omega \rangle}| |\widehat{f}(\omega)| = |\widehat{f}(\omega)|$. Esto demuestra que el operador $\Phi(f) = |\widehat{f}|$ es invariante por traslaciones. \square

No obstante, solo asegurar la invarianza por traslaciones no es adecuado. También es crucial que nuestro operador mantenga su invarianza cuando se enfrenta a pequeñas deformaciones o difeomorfismos. Por lo tanto, un operador $\Phi(f)$ se considerará estable ante deformaciones si cumple con la [Definición 1.6](#).

Sin embargo, como ya hemos comentado anteriormente, las deformaciones conducen a inestabilidades bien conocidas en altas frecuencias [[Hö71](#)]. Esto se ilustra con un pequeño operador de escala, $(T_\tau f)(x) = f(x - \tau(x)) = f((1 - \epsilon)x)$, para $\tau(x) = \epsilon x$, con $0 < \epsilon \ll 1$, $\|(\text{Jac}(\tau))(x)\|_\infty = \epsilon$ y $\|(\text{Hess}(\tau))(x)\|_\infty = 0$. Con esto, la condición de Lipschitz para el módulo de la transformada de Fourier nos daría la existencia de una constante $c > 0$ de modo que la desigualdad que se tendría que cumplir para cada $f \in L^2(\mathbb{R}^d)$ y cada $0 < \epsilon \ll 1$ sería:

$$|| |\widehat{f}| - |\widehat{T_\tau f}| || \leq c \|f\| (\| \text{Jac}(\tau) \|_\infty + \| \text{Hess}(\tau) \|_\infty) = c \|f\| \epsilon. \quad (2.1)$$

Esto implica encontrar una constante $c > 0$ que satisfaga la desigualdad para cualquier valor de ϵ . A continuación, veremos un ejemplo sencillo con una función unidimensional para mayor claridad. Si $f(x) = e^{i\langle x, \xi \rangle} \Theta(x)$, entonces escalar por $1 - \epsilon$ traslada la frecuencia central ξ a $(1 - \epsilon)\xi$. Si Θ es regular con un decaimiento rápido (es decir, tiende a cero rápidamente a medida que x se aleja del origen), entonces:

$$|| |\widehat{f}| - |\widehat{T_\tau f}| || \sim |\xi| ||\Theta|| \epsilon = |\xi| \|f\| \epsilon.$$

Dado que $|\xi|$ puede ser arbitrariamente grande, $\Phi(f) = |\widehat{f}|$ no satisface la condición de continuidad de Lipschitz al escalar frecuencias altas, ya que no es posible encontrar una constante $c > 0$ tal que la desigualdad (2.1) se cumpla para cualquier valor de ϵ .

Para abordar el problema del módulo de la transformada de Fourier, sustituiremos las ondas sinusoidales por funciones localizadas que presenten un soporte más amplio en las altas frecuencias, lo que resultará más eficaz para nuestros objetivos. Estas funciones son conocidas como ondículas.

2. Teoría de Mallat sobre Redes de Dispersión

2.1.2. Ondículas como alternativa a la transformada de Fourier

Las ondículas (*wavelets* en inglés) son funciones matemáticas que permiten descomponer señales en diferentes niveles de detalle y resolución, ofreciendo una representación localizada tanto en tiempo como en frecuencia. A diferencia de las ondas sinusoidales utilizadas en la transformada de Fourier, las ondículas son de duración finita y pueden adaptarse mejor a cambios locales de una señal, es decir, permanecen estables frente a pequeñas deformaciones. Cada ondícula se define por una función madre que se desplaza y escala para cubrir la totalidad de la señal, proporcionando una descomposición jerárquica que es útil para detectar detalles específicos y discontinuidades en las señales y las imágenes. Esto hace que sean especialmente útiles para la compresión de imágenes, el análisis de texturas y la detección de bordes. Introduciremos la transformada de ondículas y veremos cómo, a través de convoluciones con bases de ondículas, se obtienen coeficientes que son robustos ante la acción de difeomorfismos.

A diferencia de las bases de Fourier, las bases de ondículas ofrecen representaciones dispersas de señales con regiones que son solo parcialmente regulares, permitiendo capturar características como transiciones y singularidades. En imágenes, los coeficientes más grandes de las ondículas se concentran alrededor de zonas con esquinas o texturas irregulares.

Como ejemplo, consideremos la base de Haar, que aunque no será utilizada para construir nuestro propagador de dispersión, puede servir para ilustrar mejor el concepto de las ondículas. La explicación que sigue está inspirada en [Malo8].

La base de Haar se define a partir de la siguiente función:

$$\psi(t) = \begin{cases} 1 & \text{si } 0 \leq t < 1/2 \\ -1 & \text{si } 1/2 \leq t < 1. \\ 0 & \text{en otro caso} \end{cases}$$

Podemos visualizar una representación gráfica de esta en [Figura 2.1](#).

Denominamos a esta ondícula como ondícula madre, ya que, a partir de ella, es posible generar la base ortonormal

$$\left\{ \psi_{j,n}(t) = \frac{1}{\sqrt{2^j}} \psi\left(\frac{t - 2^j n}{2^j}\right) \right\}_{(j,n) \in \mathbb{Z}^2}$$

del espacio $L^2(\mathbb{R})$ de señales con energía finita (es decir, señales cuya integral del cuadrado de su valor absoluto es finita).

De este modo, cualquier señal f con energía finita puede ser expresada mediante los coeficientes que se obtienen al calcular el producto interno en $L^2(\mathbb{R})$ con la base mencionada anteriormente:

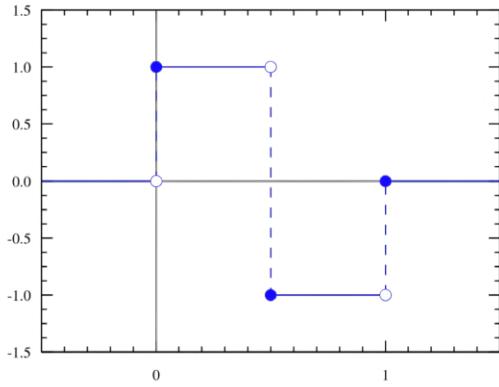


Figura 2.1.: Representación gráfica de la función base de Haar. Imagen extraída de [Wik24b].

$$\langle f, \psi_{j,n} \rangle = \int_{-\infty}^{\infty} f(t) \psi_{j,n}(t) dt.$$

Y así, dicha señal puede reconstruirse realizando la suma sobre su base ortonormal:

$$f = \sum_{j=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \langle f, \psi_{j,n} \rangle \psi_{j,n}.$$

Esto nos permite, al igual que con el módulo de la transformada de Fourier, trabajar en un dominio más simple donde es posible procesar la información de forma más eficiente, y luego reconstruir la señal a partir de los coeficientes sin perder ningún detalle.

Una de las características de la base de Haar es que sus coeficientes más altos se concentran en las zonas donde hay cambios abruptos de la señal, como bordes, esquinas o texturas en imágenes. Esto ocurre porque en esos casos es posible encontrar un elemento de la base de Haar cuyo soporte coincide con el intervalo donde ocurre el cambio, generando así un coeficiente de ondícula diferente de cero.

En el caso específico de las imágenes, las bases ortonormales de ondículas pueden ser construidas a partir de las correspondientes bases ortonormales en señales unidimensionales. Para capturar las variaciones en la imagen, utilizaremos tres ondículas, cada una diseñada para identificar los cambios en direcciones horizontales, verticales y diagonales de la imagen. Así, si denominamos a las ondículas como $\psi^1(x)$, $\psi^2(x)$ y $\psi^3(x)$, con $x \in \mathbb{R}^2$, siendo estas dilatadas por un factor 2^j , con $j \in \mathbb{Z}$, y trasladadas por $2^j n$, con $n \in \mathbb{Z}^2$, entonces podemos construir una base ortonormal para el espacio $L^2(\mathbb{R})$:

$$\left\{ \psi_{j,n}^k(x) = \frac{1}{\sqrt{2^j}} \psi^k \left(\frac{x - 2^j n}{2^j} \right) \right\}_{(j,n) \in \mathbb{Z} \times \mathbb{Z}^2, 1 \leq k \leq 3}.$$

2. Teoría de Mallat sobre Redes de Dispersión

El soporte de la ondícula $\psi_{j,n}^k(x)$ tiene la forma de un cuadrado cuya dimensión es proporcional a la escala 2^j (véase Figura 2.2). Para representar imágenes con N píxeles, las bases de ondículas bidimensionales se discretizan y definen una base ortonormal.

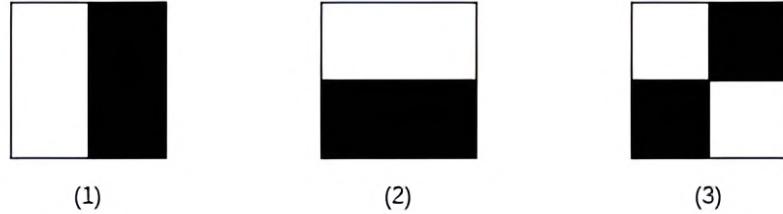


Figura 2.2.: Filtro generado por el soporte de una ondícula para detectar bordes (1) verticales, (2) horizontales y (3) diagonales. Imagen elaborada por el autor.

Al igual que en el caso unidimensional, los coeficientes de las ondículas $\langle f, \psi_{j,n}^k \rangle$ serán pequeños si $f(x)$ es regular, y serán grandes cerca de los cambios abruptos de frecuencia, como en los bordes o esquinas de las imágenes (véase Figura 2.3).

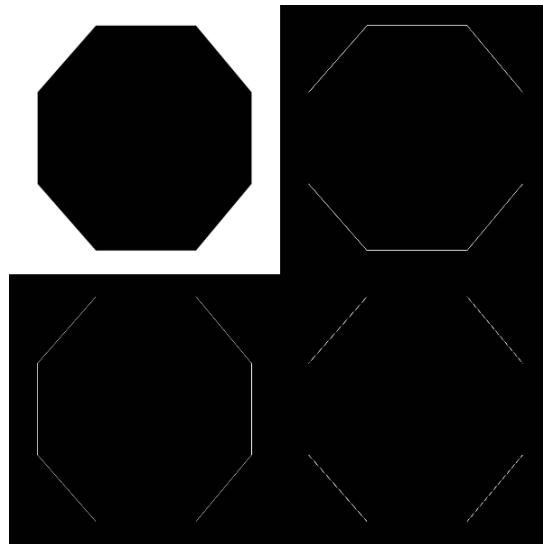


Figura 2.3.: Ejemplo de aplicación de la base de Haar a una imagen (arriba izquierda). Los filtros destacan los bordes en tres direcciones: horizontal (arriba derecha), vertical (abajo izquierda) y diagonal (abajo derecha). Imagen extraída de [Jor24].

Para avanzar en la definición del propagador de dispersión, la elección de la ondícula madre y la base ortogonal asociada estarán sujetas a cambios por escalados y rotaciones. Por lo

tanto, introducimos la siguiente definición:

Definición 2.3. Una ondícula madre escalada por un factor 2^j , con $j \in \mathbb{Z}$, y rotada por $r \in G$, siendo G el grupo finito de rotaciones, se define como:

$$\psi_{2^j r}(x) = 2^j \psi(2^j r^{-1} x).$$

Su transformada de Fourier es:

$$\widehat{\psi}_{2^j r}(\omega) = \widehat{\psi}(2^{-j} r^{-1} \omega).$$

La transformada de dispersión que utilizaremos contará con una base de ondículas generada por una ondícula madre del siguiente tipo:

$$\psi(x) = e^{i\langle x, \eta \rangle} \Theta(x),$$

donde $\Theta(x)$ es una función real con soporte en una bola de baja frecuencia centrada en $x = 0$, con radio del orden de π .

La transformada de Fourier de esta función se puede escribir como:

$$\widehat{\psi}(\omega) = \int_{\mathbb{R}^d} e^{i\langle x, \eta \rangle} \Theta(x) e^{-2\pi i\langle x, \omega \rangle} dx = \int_{\mathbb{R}^d} \Theta(x) e^{-2\pi i\langle x, \omega - \eta \rangle} dx = \widehat{\Theta}(\omega - \eta),$$

lo que significa que $\widehat{\psi}(\omega)$ está localizada en una bola con el mismo radio, pero centrada en $\omega = \eta$. Tras el escalado y la rotación, esto resulta en:

$$\widehat{\psi}_\lambda(\omega) = \widehat{\Theta}(\lambda^{-1}\omega - \eta),$$

donde $\lambda = 2^j r \in 2^{\mathbb{Z}} \times G$. En consecuencia, $\widehat{\psi}_\lambda(\omega)$ tiene soporte en una bola centrada en $\lambda\eta$ con radio proporcional a $|\lambda| = 2^j$.

2.1.3. La transformada de Littlewood-Paley

Después de conocer más a fondo las ondículas y su funcionamiento, presentamos la *transformada de ondículas de Littlewood-Paley*, que utilizaremos para construir el propagador de dispersión. Su expresión es la siguiente:

$$\forall x \in \mathbb{R}^d, \quad (W[\lambda]f)(x) = (f * \psi_\lambda)(x) = \int_{\mathbb{R}^d} f(u) \psi_\lambda(x - u) du,$$

2. Teoría de Mallat sobre Redes de Dispersión

donde $*$ denota la operación de convolución.

Calculamos ahora su transformada de Fourier, teniendo en cuenta el *teorema de convolución de la transformada de Fourier*:

Teorema 2.1. Sean f y g dos funciones integrables. Si:

$$h(x) = (f * g)(x) = \int f(y)g(x-y)dy,$$

entonces:

$$\widehat{h}(\omega) = \widehat{f}(\omega)\widehat{g}(\omega),$$

y:

$$h(x) = (f * g)(x) = \int \widehat{f}(\omega)\widehat{g}(\omega)e^{-2\pi i\langle x,\omega \rangle}d\omega.$$

De este modo, obtenemos que:

$$\widehat{(W[\lambda]f)}(\omega) = \widehat{f}(\omega)\widehat{\psi}_\lambda(\omega) = \widehat{f}(\omega)\widehat{\psi}(\lambda^{-1}\omega).$$

Cabe señalar que este resultado se utilizará de ahora en adelante en el desarrollo de este trabajo, sin necesidad de hacer mención explícita al mismo.

Considerando que, si f es real, entonces su transformada coincide con el conjugado complejo $\widehat{f}(-\omega) = \overline{\widehat{f}(\omega)}$, podemos observar que:

- Si $\widehat{\psi}(\omega)$ y f son reales, entonces $W[-\lambda]f = \overline{W[\lambda]f}$. Además, al denominar por G^+ el cociente de G sobre $\{-1, 1\}$, donde r y $-r$ son equivalentes, basta con calcular $W[2^j r]f$ para las rotaciones positivas de G^+ .
- En caso de que f sea compleja, se necesitaría calcular $W[2^j r]f$ para todo $r \in G$.

Podemos comprender que, a menor sea el factor de escala que afecta a la ondícula, más comprimida estará, y viceversa. Esto nos permite establecer una relación entre la frecuencia y la escala (véase [Figura 2.4](#)):

- Una escala pequeña implica una ondícula más comprimida, lo que permite detectar rápidamente los cambios en la señal, resultando en frecuencias más altas.
- En cambio, una escala mayor produce una ondícula más dilatada, donde solo se detectarán los cambios más abruptos, y las frecuencias resultantes serán más bajas.



Figura 2.4.: Como se observa en el caso de la izquierda, la ondícula está afectada por una escala más pequeña, lo que le permite detectar cambios en la señal con mayor precisión a frecuencias más altas a lo largo del tiempo mediante la convolución. En contraste, en el caso de la derecha, la ondícula está sujeta a una escala mayor, lo que reduce su capacidad para detectar con precisión las variaciones en la señal. Imagen extraída de [Mat24a].

Con la transformada de Littlewood-Paley sucede algo similar: a una cierta escala 2^J (con $J \in \mathbb{Z}$ fijo), solo se conservan las ondículas afectadas por un factor de escala $2^j > 2^{-J}$. Esto genera un umbral a partir del cual la base ortonormal de ondículas no podría identificar cambios de frecuencia. Por lo tanto, surge la necesidad de promediar las bajas frecuencias que no son captadas por estas ondículas en un dominio proporcional al factor 2^j :

$$A_J f = f * \phi_{2^J}, \quad \text{con} \quad \phi_{2^J}(x) = 2^{-J} \phi(2^{-J}x),$$

donde ϕ es una función fija, de la cual dependerá la transformada de ondículas.

Si f es una función real, la transformada de ondículas se expresa de la siguiente forma:

$$W_J f = \{A_J f, (W[\lambda]f)_{\lambda \in \Lambda_J}\},$$

es decir, se compone del promedio de todas las ondículas de la base que no tienen soporte en la escala fija 2^J , junto con el conjunto de coeficientes resultantes de convolucionar cada elemento de la base con la señal f para escalas $2^j > 2^{-J}$. Para esto, indexamos por $\Lambda_J = \{\lambda = 2^j r : r \in G^+, 2^j > 2^{-J}\}$.

Su norma sería:

$$\|W_J f\|^2 = \|A_J f\|^2 + \sum_{\lambda \in \Lambda_J} \|W[\lambda]f\|^2. \quad (2.2)$$

Si $J = \infty$, entonces todas las ondículas de la base tendrían coeficientes distintos de cero, por lo que:

2. Teoría de Mallat sobre Redes de Dispersión

$$W_\infty f = \{W[\lambda]f\}_{\lambda \in \Lambda_\infty},$$

con $\Lambda_\infty = 2^\mathbb{Z} \times G^+$.

La norma en este caso sería:

$$\|W_\infty f\|^2 = \sum_{\lambda \in \Lambda_\infty} \|W[\lambda]f\|^2.$$

Si f es compleja, se incluyen todas las rotaciones, obteniendo $W_J f = \{A_J f, (W[\lambda]f)_{\lambda, -\lambda \in \Lambda_J}\}$ y $W_\infty f = \{W[\lambda]f\}_{\lambda, -\lambda \in \Lambda_\infty}$.

Introducimos ahora el concepto de operador unitario:

Definición 2.4. Sea \mathcal{H} un espacio de Hilbert. Un operador $\Phi : \mathcal{H} \rightarrow \mathcal{H}$ se dice que es *unitario* si satisface las siguientes propiedades:

- Para cualquier función $f \in \mathcal{H}$, se cumple que $\|\Phi(f)\| = \|f\|$ (preservación de la norma).
- Para cualquier par de funciones $f, g \in \mathcal{H}$, se cumple que $\langle \Phi(f), \Phi(g) \rangle = \langle f, g \rangle$ (preservación del producto escalar).

La siguiente proposición establece una condición estándar de Littlewood-Paley para que W_J sea unitario:

Proposición 2.1. *Para cualquier $J \in \mathbb{Z}$ o $J = \infty$, W_J es unitario en el espacio de funciones reales o complejas de $L^2(\mathbb{R}^d)$ si, y solo si, para casi todo $\omega \in \mathbb{R}^d$ se cumple:*

$$\beta \sum_{j=-\infty}^{\infty} \sum_{r \in G} |\widehat{\psi}(2^{-j}r^{-1}\omega)|^2 = 1 \quad y \quad |\widehat{\phi}(\omega)|^2 = \beta \sum_{j=-\infty}^0 \sum_{r \in G} |\widehat{\psi}(2^{-j}r^{-1}\omega)|^2, \quad (2.3)$$

donde $\beta = 1$ para funciones complejas y $\beta = \frac{1}{2}$ para funciones reales.

Demostración. Si f es una función compleja, entonces tenemos que $\beta = 1$, y vamos a demostrar que (2.3) es equivalente a:

$$\forall J \in \mathbb{Z}, \quad |\widehat{\phi}(2^J\omega)|^2 + \sum_{j>-J, r \in G} |\widehat{\psi}(2^{-j}r^{-1}\omega)|^2 = 1. \quad (2.4)$$

Partiendo de (2.3) con $\beta = 1$, se obtiene:

$$\sum_{j=-\infty}^{\infty} \sum_{r \in G} |\widehat{\psi}(2^{-j}r^{-1}\omega)|^2 = 1 \quad \text{y} \quad |\widehat{\phi}(\omega)|^2 = \sum_{j=-\infty}^0 \sum_{r \in G} |\widehat{\psi}(2^{-j}r^{-1}\omega)|^2.$$

Sumando $\sum_{j=0}^{\infty} \sum_{r \in G} |\widehat{\psi}(2^{-j}r^{-1}\omega)|^2$ en la segunda ecuación, obtenemos:

$$|\widehat{\phi}(\omega)|^2 + \sum_{j=0}^{\infty} \sum_{r \in G} |\widehat{\psi}(2^{-j}r^{-1}\omega)|^2 = 1.$$

Por otro lado, si nos fijamos en la expresión (2.4), a la cual queremos llegar, se tiene que:

$$\begin{aligned} \forall J \in \mathbb{Z}, \quad & |\widehat{\phi}(2^J \omega)|^2 + \sum_{j>-J, r \in G} |\widehat{\psi}(2^{-j}r^{-1}\omega)|^2 = 1 \\ \iff \forall J \in \mathbb{Z}, \quad & |\widehat{\phi}(2^J \omega)|^2 = \sum_{j=-\infty}^{-J} \sum_{r \in G} |\widehat{\psi}(2^{-j}r^{-1}\omega)|^2. \end{aligned}$$

Con lo cual, si demostramos esta última igualdad, tendremos que (2.3) y (2.4) son equivalentes para el caso $\beta = 1$. Así:

$$|\widehat{\phi}(2^J \omega)|^2 = \sum_{j=-\infty}^0 \sum_{r \in G} |\widehat{\psi}(2^{-j}r^{-1}2^J \omega)|^2 = \sum_{j=-\infty}^0 \sum_{r \in G} |\widehat{\psi}(2^{J-j}r^{-1}\omega)|^2 = \sum_{j=-\infty}^{-J} \sum_{r \in G} |\widehat{\psi}(2^{-j}r^{-1}\omega)|^2.$$

De esta manera, queda demostrado que las expresiones (2.3) y (2.4) son equivalentes para el caso $\beta = 1$.

Como $(W[2^j r] f)(\omega) = \widehat{f}(\omega) \widehat{\psi}_{2^j r}(\omega)$, entonces multiplicando la ecuación (2.4) por $|\widehat{f}(\omega)|^2$, obtenemos:

$$\forall J \in \mathbb{Z}, \quad |\widehat{\phi}(2^J \omega)|^2 |\widehat{f}(\omega)|^2 + \sum_{j>-J, r \in G} |\widehat{\psi}(2^{-j}r^{-1}\omega)|^2 |\widehat{f}(\omega)|^2 = |\widehat{f}(\omega)|^2.$$

Si ahora integramos en ambos miembros sobre \mathbb{R}^d , tenemos que:

$$\int_{\mathbb{R}^d} \left(|\widehat{\phi}(2^J \omega)|^2 |\widehat{f}(\omega)|^2 + \sum_{j>-J, r \in G} |\widehat{\psi}(2^{-j}r^{-1}\omega)|^2 |\widehat{f}(\omega)|^2 \right) d\omega = \int_{\mathbb{R}^d} |\widehat{f}(\omega)|^2 d\omega.$$

Aplicando la *fórmula de Plancherel*, que nos indica que para el caso de \mathbb{R}^d se cumple la igualdad $\int_{\mathbb{R}^d} |f(x)|^2 dx = \int_{\mathbb{R}^d} |\widehat{f}(\omega)|^2 d\omega$, obtenemos:

$$\int_{\mathbb{R}^d} \left(|\phi(2^J x)|^2 |f(x)|^2 + \sum_{j>-J, r \in G} |\psi(2^{-j} r^{-1} x)|^2 |f(x)|^2 \right) dx = \int_{\mathbb{R}^d} |f(x)|^2 dx.$$

Si ahora tomamos en cuenta la ecuación (2.2), obtenemos que la expresión anterior es equivalente a:

$$\|A_J f\|^2 + \sum_{\lambda \in \Lambda_J} \|W[\lambda]f\|^2 = \|W_J f\|^2 = \|f\|^2,$$

que es válido para todo J , y en particular también para $J = \infty$.

Recíprocamente, si $\|W_J f\|^2 = \|f\|^2$, entonces (2.4) se cumple para casi todo ω . De no ser así, podríamos construir una función f no nula cuya transformada de Fourier \widehat{f} tuviera soporte en una región de ω donde (2.4) no se verificaría. En ese caso, al aplicar la fórmula de Plancherel, obtendríamos que $\|W_J f\|^2 \neq \|f\|^2$, lo cual contradice la hipótesis. Dado que (2.4) es equivalente a lo demostrado anteriormente, queda probado el resultado para el caso en que f es compleja.

Si f es una función real, entonces $|\widehat{f}(\omega)| = |\widehat{f}(-\omega)|$, lo que implica que $\|W[2^j r]f\| = \|W[-2^j r]f\|$. Por lo tanto, $\|W_J f\|$ permanece constante si restringimos r a G^+ , y al multiplicar ψ por $\sqrt{2}$ obtenemos la condición (2.3) con $\beta = \frac{1}{2}$.

□

2.1.4. Convenciones para futuras secciones

Llegados a este punto, ya contamos con la transformada de ondículas que utilizaremos para la construcción del propagador de dispersión. A continuación, definimos algunas propiedades que impondremos a los elementos que la componen y que usaremos de ahora en adelante:

- $\widehat{\psi}$ es una función real que satisface la condición (2.3), lo que implica que $\widehat{\psi}(0) = \int \psi(x)dx = 0$ y $|\widehat{\psi}(r\omega)| = |\widehat{\psi}(\omega)|$, $\forall r \in G$.
- $\widehat{\phi}(\omega)$ es real y simétrica, por lo que ϕ también lo es, y $\phi(rx) = \phi(x)$, $\forall r \in G$.
- Las derivadas de ϕ pertenecen a $L^1(\mathbb{R}^d)$.

A continuación, introducimos algo de notación que utilizaremos de ahora en adelante:

2.2. El operador de dispersión sobre un camino ordenado

- Denotaremos por $f^\lambda(x) := e^{-i\lambda\langle x, \eta \rangle} f(x)$ a la señal $f(x)$ modulada en frecuencia por $-\lambda\eta$.
- Se denota $(g \circ f)(x) := f(gx)$ a la acción de un elemento del grupo $g \in G$.
- Un operador R parametrizado por p es denotado por $R[p]$ y $R[\Omega] := \{R[p]\}_{p \in \Omega}$.

Si f se escala y rota, $((2^l g) \circ f)(x) = f(2^l gx)$, con $2^l g \in 2^{\mathbb{Z}} \times G$, entonces la transformada de ondículas se escala y rota de acuerdo a:

$$\begin{aligned}
 (W[\lambda]((2^l g) \circ f))(x) &= (((2^l g) \circ f) * \psi_\lambda)(x) = \int_{\mathbb{R}^d} ((2^l g) \circ f)(u) \psi_\lambda(x - u) du \\
 &= \int_{\mathbb{R}^d} f(2^l gu) \psi_\lambda(x - u) du \\
 &= \int_{\mathbb{R}^d} f(v) \psi_\lambda(x - 2^{-l} g^{-1} v) 2^{-l} dv \\
 &= \int_{\mathbb{R}^d} f(v) 2^j \psi(2^j r^{-1}(x - 2^{-l} g^{-1} v)) 2^{-l} dv \\
 &= \int_{\mathbb{R}^d} f(v) 2^{j-l} \psi(2^j r^{-1} 2^{-l} g^{-1}(2^l gx - v)) dv \\
 &= \int_{\mathbb{R}^d} f(v) 2^{j-l} \psi(2^{j-l}(rg)^{-1}(2^l gx - v)) dv \\
 &= \int_{\mathbb{R}^d} f(v) \psi_{2^{-l} g \lambda}(2^l gx - v) dv \\
 &= (f * \psi_{2^{-l} g \lambda})(2^l gx) = (W[2^{-l} g \lambda]f)(2^l gx) \\
 &= ((2^l g) \circ (W[2^{-l} g \lambda]f))(x),
 \end{aligned} \tag{2.5}$$

donde en (2.5) se ha realizado el cambio de variable $v = 2^l gu$, de forma que $\det(\text{Jac}(u)) = \det(\text{Jac}(2^{-l} g^{-1} v)) = 2^{-l}$.

Como ϕ es invariante a rotaciones en G , podemos comprobar que A_J commuta con las rotaciones de G :

$$A_J(g \circ f) = g \circ (A_J f), \quad \forall g \in G.$$

2.2. El operador de dispersión sobre un camino ordenado

2.2.1. Obtención de coeficientes invariantes por traslaciones

La transformada de Littlewood-Paley discutida anteriormente cumple con la propiedad de ser Lipschitz-continua bajo difeomorfismos, ya que las ondículas que emplea están localizadas.

2. Teoría de Mallat sobre Redes de Dispersión

das y son regulares. No obstante, todavía no es invariante a traslaciones, pues cuando se traslada f , también se traslada $W[\lambda]f = f * \psi_\lambda$. El siguiente objetivo es calcular coeficientes que permanezcan invariables frente a traslaciones, que permanezcan estables bajo la acción de difeomorfismos y que retengan la información en altas frecuencias que proporcionan las ondículas. Reuniendo todas estas propiedades, podemos conformar el operador adecuado que permitirá la construcción del propagador de dispersión.

Los coeficientes que se mantienen invariables bajo traslaciones serán obtenidos a través de la acción de un operador no-lineal, lo cual nos lleva al siguiente lema:

Lema 2.2. *Si $U[\lambda]$ es un operador definido en $L^2(\mathbb{R}^d)$, no necesariamente lineal pero que commuta con traslaciones, entonces $\int_{\mathbb{R}^d} (U[\lambda]f)(x)dx$ es invariante a traslaciones si es finito.*

Demostración. Sea $f \in L^2(\mathbb{R}^d)$, $t \in \mathbb{R}^d$ y $(T_t f)(x) = f(x - t)$ una traslación de f . Como $U[\lambda]f$ commuta con traslaciones, se tiene que:

$$(U[\lambda](T_t f))(x) = U[\lambda](f(x - t)) = (U[\lambda]f)(x - t) = (T_t(U[\lambda]f))(x).$$

Vamos a comprobar ahora que si $\int_{\mathbb{R}^d} (U[\lambda]f)(x)dx$ es finito, entonces la integral es invariante a traslaciones. En otras palabras, queremos comprobar que:

$$\int_{\mathbb{R}^d} (U[\lambda](T_t f))(x)dx = \int_{\mathbb{R}^d} (U[\lambda]f)(x)dx.$$

Para ello, si tenemos en cuenta la commutatividad del operador $U[\lambda]$, se tiene que:

$$\int_{\mathbb{R}^d} (U[\lambda](T_t f))(x)dx = \int_{\mathbb{R}^d} U[\lambda](f(x - t))dx = \int_{\mathbb{R}^d} (U[\lambda]f)(x - t)dx.$$

Y tras esto, basta tener en cuenta el cambio de variable $y = x - t$, con Jacobiano igual a uno, y se tendría en la expresión anterior:

$$\int_{\mathbb{R}^d} (U[\lambda]f)(x - t)dx = \int_{\mathbb{R}^d} (U[\lambda]f)(y)dy.$$

Por lo que la integral es invariante por traslaciones. □

En nuestro caso, $W[\lambda]f = f * \psi_\lambda$ es un ejemplo trivial de este lema, ya que es un operador que commuta con traslaciones y $\int_{\mathbb{R}^d} (f * \psi)(x)dx = 0$, debido a que $\int_{\mathbb{R}^d} \psi(x)dx = 0$.

Para lograr un operador no trivial $U[\lambda]f$ que sea invariante a traslaciones, necesitamos combinar $W[\lambda]$ con un operador no-lineal adicional $M[\lambda]$, llamado *operador de demodulación*. Este

operador de demodulación convierte $W[\lambda]f$ en una función de frecuencia más baja cuya integral es diferente de cero. Además, la elección de $M[\lambda]$ debe preservar la continuidad de Lipschitz bajo la acción de los difeomorfismos. En resumen, buscamos un operador no-lineal que produzca coeficientes invariantes bajo traslaciones, no triviales, y que también conserve la Lipschitz-continuidad.

A continuación, mostramos un ejemplo para ilustrar mejor este concepto.

Si la ondícula madre es de la forma $\psi(x) = e^{i\langle x, \eta \rangle} \Theta(x)$, entonces los elementos de la base son $\psi_\lambda(x) = e^{i\lambda\langle x, \eta \rangle} \Theta_\lambda(x)$, por lo tanto:

$$(W[\lambda]f)(x) = f * \psi_\lambda(x) = (f * e^{i\lambda\langle x, \eta \rangle} \Theta_\lambda)(x) = e^{i\lambda\langle x, \eta \rangle} ((f^\lambda * \Theta_\lambda)(x)). \quad (2.6)$$

Para lograr un operador invariante a traslaciones, es posible eliminar el término de modulación $e^{i\lambda\langle x, \eta \rangle}$ mediante una función $M[\lambda]$. Por ejemplo:

$$(M[\lambda]h)(x) = e^{-i\lambda\langle x, \eta \rangle} e^{-i\Phi(\widehat{h}(\lambda\eta))} h(x),$$

donde $\Phi(\widehat{h}(\lambda\eta))$ es la fase compleja de $\widehat{h}(\lambda\eta)$. Este registro de fase no-lineal asegura que $M[\lambda]$ permanece invariante bajo traslaciones, dado que:

$$\begin{aligned} \int_{\mathbb{R}^d} (M[\lambda](W[\lambda]f))(x) dx &= \int_{\mathbb{R}^d} e^{-i\lambda\langle x, \eta \rangle} e^{-i\Phi(\widehat{W[\lambda\eta]f})} \left(e^{i\lambda\langle x, \eta \rangle} \left(((e^{-i\lambda\langle x, \eta \rangle} f) * \Theta_\lambda)(x) \right) \right) dx \\ &= e^{-i\Phi(\widehat{f}(\lambda\eta)\widehat{\psi}_\lambda(\lambda\eta))} \int_{\mathbb{R}^d} ((e^{-i\lambda\langle x, \eta \rangle} f) * \Theta_\lambda)(x) dx \\ &= e^{-i\Phi(\widehat{f}(\lambda\eta)\widehat{\psi}_\lambda(\lambda\eta))} ((e^{-i\lambda\langle x, \eta \rangle} f) * \Theta_\lambda)(0) \\ &= e^{-i\Phi(\widehat{f}(\lambda\eta)\widehat{\psi}_\lambda(\lambda\eta))} \cdot (e^{-i\lambda\langle x, \eta \rangle} f)(0) \cdot \widehat{\Theta}_\lambda(0) \\ &= e^{-i\Phi(\widehat{f}(\lambda\eta)\widehat{\psi}_\lambda(\lambda\eta))} \cdot \widehat{f}(\lambda\eta) \cdot \widehat{\Theta}_\lambda(0) \\ &= \left| \widehat{f}(\lambda\eta) \cdot \widehat{\Theta}_\lambda(0) \right| \\ &= \left| \widehat{f}(\lambda\eta) \right| \left| \widehat{\Theta}_\lambda(0) \right| \\ &= \left| \widehat{f}(\lambda\eta) \right| \left| \widehat{\Theta}(0) \right|. \end{aligned}$$

Como podemos observar, la integral resulta ser no trivial y, por otro lado, obtenemos el módulo de la transformada de Fourier que, como se discutió en el [Lema 2.1](#), es invariante por traslaciones. Sin embargo, este operador no será utilizado para nuestros fines, ya que, además de ser complejo, no cumple con las condiciones de Lipschitz-continuidad bajo difeomorfismos como ya vimos anteriormente, lo que resulta en inestabilidades bien conocidas en altas frecuencias.

2. Teoría de Mallat sobre Redes de Dispersión

2.2.2. El operador módulo

En nuestro caso, para garantizar la Lipschitz-continuidad bajo difeomorfismos, es necesario que $M[\lambda]$ conmute con estos, y que además sea no-expansivo, asegurando así la estabilidad en $L^2(\mathbb{R}^d)$. Se demuestra que, en tales condiciones, $M[\lambda]$ debe ser un operador definido punto a punto (es decir, un operador que actúa aplicando su definición sobre los valores de la función de manera independiente para cada punto del dominio, sin considerar el comportamiento en puntos vecinos) [Bru12], lo que implica que el operador $(M[\lambda]h)(x)$ que buscamos dependería únicamente del valor de h en el punto x .

Para obtener mejores propiedades, vamos a imponer que $\|M[\lambda]h\| = \|h\|$, $\forall h \in L^2(\mathbb{R}^d)$, lo que implica que $|M[\lambda]h| = |h|$, ya que:

$$\begin{aligned} \|M[\lambda]h\| = \|h\| &\iff \left(\int_{\mathbb{R}^d} |(M[\lambda]h)(x)|^2 dx \right)^{\frac{1}{2}} = \left(\int_{\mathbb{R}^d} |h(x)|^2 dx \right)^{\frac{1}{2}} \\ &\iff \int_{\mathbb{R}^d} |(M[\lambda]h)(x)|^2 dx = \int_{\mathbb{R}^d} |h(x)|^2 dx \\ &\iff \int_{\mathbb{R}^d} \left(|(M[\lambda]h)(x)|^2 - |h(x)|^2 \right) dx = 0 \\ &\iff |(M[\lambda]h)(x)|^2 - |h(x)|^2 = 0 \\ &\iff |(M[\lambda]h)(x)| = |h(x)|. \end{aligned}$$

Para alcanzar la conclusión sugerida por el autor [Mal12], se ha supuesto que $|(M[\lambda]h)(x)| \geq |h(x)|$, $\forall x \in \mathbb{R}^d$, lo que junto con el hecho de que ambas funciones en el integrando son positivas, nos da el resultado esperado.

Con el fin de cumplir con todas las restricciones establecidas, aplicaremos el operador $M[\lambda]h = |h|$, el cual elimina cualquier variación en la fase, tal como se indica en [BM13]. Así, de acuerdo con la ecuación (2.6), este módulo convierte $W[\lambda]f$ en una señal con una frecuencia más baja que la original:

$$M[\lambda](W[\lambda]f) = |W[\lambda]f| = |f^\lambda * \Theta_\lambda|.$$

Vamos a ilustrar con un ejemplo cómo, cuando dos señales interfieren al aplicar este operador, la frecuencia resultante es más baja que cualquiera de las frecuencias originales.

Si $f(x) = \cos(\xi_1 x) + a \cos(\xi_2 x)$, donde $\xi_1 > 0$ y $\xi_2 > 0$ están en la banda de frecuencia de $\widehat{\psi}_\lambda$, entonces, aplicando el operador módulo obtenemos:

$$|(f * \psi_\lambda)(x)| = 2^{-1} \left| \widehat{\psi}_\lambda(\xi_1) + a \widehat{\psi}_\lambda(\xi_2) e^{i(\xi_2 - \xi_1)x} \right|,$$

que oscila en la frecuencia de interferencias $|\xi_2 - \xi_1|$, que es menor que $|\xi_1|$ y $|\xi_2|$.

De este modo, dado cómo hemos construido el operador $U[\lambda]f$, la integral $\int_{\mathbb{R}^d} (U[\lambda]f)(x)dx = \int_{\mathbb{R}^d} |(f * \psi_\lambda)(x)|dx$ se mantiene invariante por traslaciones, aunque elimina las frecuencias altas de $|f * \psi_\lambda|$. Para recuperar estas frecuencias, el propagador de dispersión calcula los coeficientes de ondículas para cada $U[\lambda]f$, es decir, $\{U[\lambda]f * \psi_{\lambda'}\}_{\lambda'}$. Nuevamente, los coeficientes invariantes por traslaciones se calculan con el módulo $U[\lambda'](U[\lambda]f) = |(U[\lambda]f) * \psi_{\lambda'}|$, e integrando $\int_{\mathbb{R}^d} (U[\lambda'](U[\lambda]f))(x)dx$.

Tomando como ejemplo el caso anterior, $f(x) = \cos(\xi_1 x) + a \cos(\xi_2 x)$, con $a < 1$, si $|\xi_2 - \xi_1| << |\lambda|$, con $|\xi_2 - \xi_1|$ dentro del soporte de $\widehat{\psi}_{\lambda'}$, entonces $U[\lambda'](U[\lambda]f)$ es proporcional a $a \cdot |\psi_\lambda(\xi_1)| \cdot |\psi_{\lambda'}(\xi_2 - \xi_1)|$. La segunda ondícula $\widehat{\psi}_{\lambda'}$ capta las interferencias creadas por el módulo, entre las frecuencias de las componentes de f y el soporte de $\widehat{\psi}_\lambda$.

2.2.3. El propagador de dispersión

Estamos ya en condiciones de definir el propagador de dispersión.

Definición 2.5. Una secuencia de elementos ordenados $p = (\lambda_1, \lambda_2, \dots, \lambda_m)$ con $\lambda_k \in \Lambda_\infty = 2^\mathbb{Z} \times G^+$ se conoce como *camino*. El camino vacío se representa por $p = \emptyset$.

Definición 2.6. Un *propagador de dispersión* es un producto por un camino ordenado de operadores no comutativos de la forma $U[p]f = U[\lambda_m](U[\lambda_{m-1}](\dots(U[\lambda_1]f))) = |f * \psi_\lambda| = |\int_{\mathbb{R}^d} f(u)\psi_\lambda(x-u)du|$, con $f \in L^2(\mathbb{R}^d)$. Es decir:

$$U[p]f = U[\lambda_m] \cdots U[\lambda_2]U[\lambda_1],$$

con $U[\emptyset] = Id$.

El operador $U[p]$ está bien definido en $L^2(\mathbb{R}^d)$ porque $\|U[\lambda]f\| = \|f\| \leq \|\psi_\lambda\|_1 \|f\|$, para todo $\lambda \in \Lambda_\infty$, donde $\|f\|_1 := \int |f(x)|dx$ denota la norma de f en $L^1(\mathbb{R}^d)$.

El propagador de dispersión es por tanto una cascada de convoluciones y módulos:

$$U[p]f = \left| \left| f * \psi_{\lambda_1} \right| * \psi_{\lambda_2} \right| \cdots \left| \psi_{\lambda_m} \right|.$$

Cada $U[\lambda]$ filtra la frecuencia del componente en la banda cubierta por $\widehat{\psi}_\lambda$ y lo aplica en un espacio de frecuencias menores utilizando el módulo.

A continuación probaremos algunas propiedades de los caminos de frecuencias tal como los describimos previamente. Para ello, comenzamos con algunas definiciones útiles:

2. Teoría de Mallat sobre Redes de Dispersión

Definición 2.7. Definimos la *rotación* y *reescalado* de un camino p por $2^l g \in 2^{\mathbb{Z}} \times G$ como $(2^l g)p = (2^l g\lambda_1, 2^l g\lambda_2, \dots, 2^l g\lambda_m)$.

Definición 2.8. La *concatenación* de dos caminos p y p' se define por:

$$p + p' = (\lambda_1, \lambda_2, \dots, \lambda_m, \lambda'_1, \lambda'_2, \dots, \lambda'_{m'}).$$

En particular, $p + \lambda = (\lambda_1, \lambda_2, \dots, \lambda_m, \lambda)$.

Con esta notación, presentamos la siguiente proposición:

Proposición 2.2. Sean p y p' dos caminos, se cumple que:

$$U[p + p'] = U[p']U[p].$$

Demostración. Como $p + p' = (\lambda_1, \lambda_2, \dots, \lambda_m, \lambda'_1, \lambda'_2, \dots, \lambda'_{m'})$, siguiendo la definición de $U[p]$, se tiene que:

$$U[p + p'] = U[\lambda'_{m'}] \dots U[\lambda'_2]U[\lambda'_1]U[\lambda_m] \dots U[\lambda_2]U[\lambda_1] = U[p']U[p].$$

□

En la [Subsección 2.1.3](#) mencionábamos que si la función f es compleja, entonces la transformada de ondículas es $W_\infty = \{W[\lambda]f\}_{\lambda, -\lambda \in \Lambda_\infty}$. Sin embargo, en este caso, debido a que el módulo $U[\lambda_1]f = |W[\lambda_1]f|$ es una función real, solo sería necesario calcular las siguientes transformadas para $\lambda_k \in \Lambda_\infty$. Por lo tanto, los propagadores de dispersión de funciones complejas se definirán en caminos positivos $p = (\lambda_1, \lambda_2, \dots, \lambda_m)$ y caminos negativos $-p = (-\lambda_1, \lambda_2, \dots, \lambda_m)$. Si f es real, entonces $W[-\lambda_1]f = W[\lambda_1]\bar{f}$, por lo que $U[-\lambda_1]f = U[\lambda_1]f$ y, por lo tanto, $U[-p]f = U[p]f$.

Con el objetivo de simplificar los cálculos, todos los resultados siguientes serán considerados sobre propagadores de dispersión, restringidos a caminos positivos, aplicados a funciones reales. Estos resultados se aplicarían de igual forma a funciones complejas incluyendo caminos negativos.

2.2.4. La transformada de dispersión

En este momento, disponemos de un operador $U[\lambda]$ que satisface todas las condiciones que deseábamos, por lo que podemos avanzar hacia la modelización matemática de una red neuronal convolucional.

Definición 2.9. Sea \mathcal{P}_∞ el conjunto de todos los caminos finitos. Definimos la *transformada de dispersión* de $f \in L^1(\mathbb{R}^d)$ para cualquier camino $p \in \mathcal{P}_\infty$ como:

$$(\bar{S}f)(p) = \int_{\mathbb{R}^d} (U[p]f)(x)dx.$$

$\bar{S}f$ es invariante por traslaciones, ya que hemos visto que el operador $U[p]$ asegura que la integral sea finita, y, por lo tanto, sea invariante por traslaciones.

Esta última definición tiene varias similitudes con el módulo de la transformada de Fourier. Sin embargo, a diferencia de la transformada de Fourier, la transformada de dispersión es Lipschitz-continua bajo la acción de los difeomorfismos, ya que se basa en iteraciones de transformadas de ondículas y módulos que, como mencionamos previamente, son estables ante la acción de dichos difeomorfismos.

No obstante, para problemas de clasificación, se suele preferir generar descriptores pequeños que sean invariantes por traslaciones frente a una escala predefinida 2^J , manteniendo las frecuencias superiores a 2^{-J} , lo que nos permite ver esta variabilidad espacial. Esto se logra convolucionando la transformada con una ventana escalada a la frecuencia deseada, en este caso $\phi_{2^J}(x) = 2^{-J}\phi(2^{-J}x)$.

Definición 2.10. Sea $J \in \mathbb{Z}$ y \mathcal{P}_J el conjunto de caminos finitos $p = (\lambda_1, \lambda_2, \dots, \lambda_m)$, con $\lambda_k \in \Lambda_J$ y $|\lambda_k| = 2^k > 2^{-J}$. Definimos una *transformada de dispersión de ventana* para todo $p \in \mathcal{P}_J$ como:

$$(S_J[p]f)(x) = ((U[p]f) * \phi_{2^J})(x) = \int_{\mathbb{R}^d} (U[p]f)(u) \phi_{2^J}(x - u) du,$$

donde la convolución con ϕ_{2^J} localiza el propagador de dispersión en dominios proporcionales a 2^J :

$$(S_J[p]f)(x) = (|| |f * \psi_{\lambda_1}| * \psi_{\lambda_2} | \cdots * \psi_{\lambda_m} | * \phi_{2^J}) (x).$$

En particular, $S_J[\emptyset]f = f * \phi_{2^J}$.

Esto define una familia infinita de funciones indexadas por \mathcal{P}_J , denotada por:

$$S_J[\mathcal{P}_J]f := \{S_J[p]f\}_{p \in \mathcal{P}_J}.$$

Si observamos, para cada camino p , $(S_J[p]f)(x)$ es una función que actúa sobre la ventana centrada en la posición x , cuyo tamaño corresponde a intervalos de dimensión 2^J . En el caso de funciones complejas, solamente necesitaríamos incluir en \mathcal{P}_J los caminos negativos, y si f es real, $S_J[-p]f = S_J[p]f$. En la Sección 2.3 se puede verificar que para ondículas adecuadas, $\|f\|^2 = \sum_{p \in \mathcal{P}_J} \|S_J[p]f\|^2$.

Sin embargo, la energía de la señal se concentra en un conjunto mucho más pequeño de caminos de frecuencias descendentes $p = (\lambda_k)_{k \leq m}$, en el cual $|\lambda_{k+1}| \leq |\lambda_k|$. Esto es debido a que el propagador $U[\lambda]$ reduce progresivamente la energía de la señal a frecuencias más bajas, hasta que en cierto punto es nula.

Ahora examinemos la relación entre este propagador de ventana y el que se definió originalmente en la Definición 2.9. Como $\phi(x)$ es continua en 0, si $f \in L^1(\mathbb{R}^d)$, se tiene que

2. Teoría de Mallat sobre Redes de Dispersión

su transformada de dispersión de ventana converge punto a punto a la transformada de dispersión cuando la escala 2^J tiende a infinito:

$$\begin{aligned}
\forall x \in \mathbb{R}^d, \quad \lim_{J \rightarrow \infty} 2^J (S_J[p]f)(x) &= \lim_{J \rightarrow \infty} 2^J ((U[p]f) * \phi_{2^J})(x) \\
&= \lim_{J \rightarrow \infty} 2^J \int_{\mathbb{R}^d} (U[p]f)(u) \phi_{2^J}(x - u) du \\
&= \lim_{J \rightarrow \infty} 2^J \int_{\mathbb{R}^d} (U[p]f)(u) 2^{-J} \phi(2^{-J}(x - u)) du \\
&= \int_{\mathbb{R}^d} (U[p]f)(u) \phi(0) du \\
&= \phi(0) \int_{\mathbb{R}^d} (U[p]f)(u) du \\
&= \phi(0) (\bar{S}f)(p).
\end{aligned}$$

Además de la convergencia puntual, es importante notar que, mientras que hasta ahora hemos trabajado exclusivamente en $L^2(\mathbb{R}^d)$, el operador aquí presentado se encuentra en $L^1(\mathbb{R}^d)$. No obstante, en la Sección 3.2 de [Mal12], se demuestra que el operador $(\bar{S}f)(p)$, $\forall f \in L^2(\mathbb{R}^d)$, y para todo camino p , pertenece a $L^2(\mathbb{R}^d)$. Además, se proporciona una condición suficiente que garantiza la convergencia uniforme del operador de dispersión de ventana S_Jf en $\bar{S}f$. Sin embargo, la demostración del recíproco no se da, aunque aparece en numerosos ejemplos. Debido a la falta de tiempo y complejidad técnica de esa sección, hemos citado dicha referencia para justificar por qué el operador está definido en $L^1(\mathbb{R}^d)$.

2.3. Propagación de la dispersión y conservación de la norma

2.3.1. Proceso iterativo del propagador de la dispersión

A partir de este punto, denotamos por $S_J[\Omega] := \{S_J[p]\}_{p \in \Omega}$ y $U[\Omega] := \{U[p]\}_{p \in \Omega}$ a la familia de operadores indexados por el conjunto de caminos $\Omega \subset \mathcal{P}_\infty$. De esta manera, un dispersor de ventanas S_J se puede calcular iterando en el propagador de un paso definido anteriormente como:

$$U_Jf = \{A_Jf, (U[\lambda]f)_{\lambda \in \Lambda_J}\},$$

donde $A_Jf = f * \phi_{2^J}$ y $U[\lambda]f = |f * \psi_\lambda|$.

Al calcular U_Jf , aplicando nuevamente U_J a cada coeficiente $U[\lambda]f$, se genera una familia infinita más extensa de funciones. Esta descomposición continua de forma recursiva aplicando U_J a cada $U[p]f$.

Con base en la [Proposición 2.2](#), se sabe que $U[\lambda]U[p] = U[p + \lambda]$, y $A_JU[p] = S_J[p]$, lo que

2.3. Propagación de la dispersión y conservación de la norma

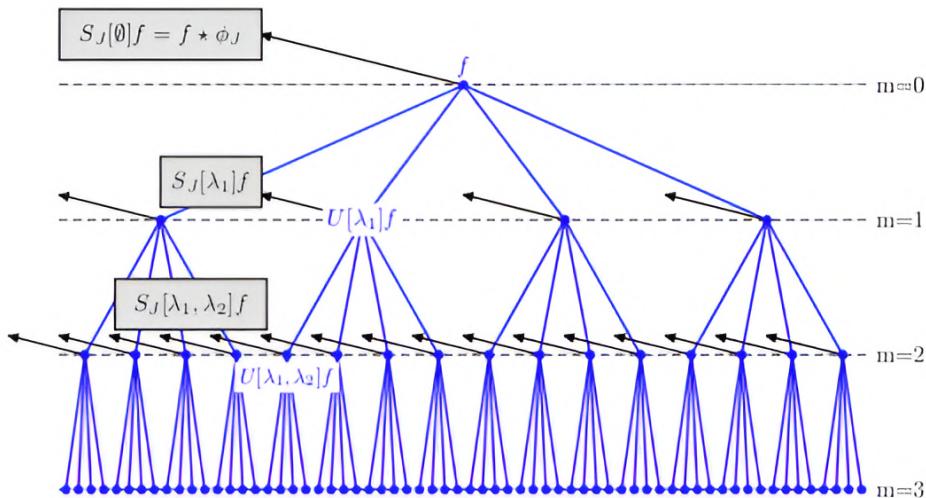
da lugar a:

$$U_J U[p] = \{S_J[p]f, (U[p + \lambda]f)_{\lambda \in \Lambda_J}\}.$$

Por lo tanto, podemos analizar el comportamiento de la transformada de dispersión de ventana según la longitud m del camino que se está utilizando. Sea Λ_J^m el conjunto de caminos de longitud m con, $\Lambda_J^0 = \emptyset$, entonces:

$$U_J(U[\Lambda_J^m]f) = \{S_J[\Lambda_J^m]f, U[\Lambda_J^{m+1}]f\}. \quad (2.7)$$

Del hecho de que $\mathcal{P}_J = \bigcup_{m \in \mathbb{N}} \Lambda_J^m$, es posible calcular $S_J[\mathcal{P}_J]f$ a partir de $f = U[\emptyset]f$, realizando iteraciones sucesivas del cálculo de $U_J(U[\Lambda_J^m]f)$ cuando m va desde 0 hasta infinito (véase [Figura 2.5](#)).



[Figura 2.5.](#): Un propagador de dispersión U_J aplicado a un punto de una señal $f(x)$ calcula $(U[\lambda_1]f)(x) = |(f * \psi_{\lambda_1})(x)|$. En la capa $m = 0$, se promedian los coeficientes que dieron 0 (por tener $2^j < 2^{-J}$), obteniendo como salida $(S_J[\emptyset]f)(x) = (f * \phi_{2^J})(x)$ (ver flecha negra). Luego, se aplica U_J a cada coeficiente $(U[\lambda_1]f)(x)$ del paso anterior, calculando así $(U[\lambda_1, \lambda_2]f)(x)$ y obteniendo como salida $(S_J[\lambda_1]f)(x) = ((U[\lambda_1]f) * \phi_{2^J})(x)$ en la capa $m = 1$. Repitiendo este proceso de forma recursiva para cada coeficiente $(U[p]f)(x)$ se va obteniendo $(S_J[p]f)(x) = ((U[p]f)(x) * \phi_{2^J})$ como salida de cada capa. Imagen extraída de [\[BM13\]](#).

2. Teoría de Mallat sobre Redes de Dispersión

2.3.2. Comparativa con una red neuronal convolucional

Las operaciones de la transformada de dispersión de ventana descritas siguen la estructura general de la red neuronal convolucional introducida por LeCun [LBH15], donde las redes neuronales convolucionales se describen como una cascada de convoluciones (la transformada de ondículas $W[\lambda]$) y funciones de activación no-lineales (el operador $M[\lambda]$), las cuales se representan en nuestro caso como módulos de números complejos. Cabe destacar que no se incluyen las capas de pooling en esta modelización.

Además, si p es un camino de longitud m , entonces a $(S_J[p]f)(x)$ se le denomina *coeficiente de orden m a escala 2^J* y es equivalente al tensor que corresponde a los mapas de activación en la red neuronal convolucional tras la convolución con el kernel de la capa m .

Sin embargo, como diferencia a destacar en esta comparativa, sabemos que las redes neuronales convolucionales han sido muy efectivas en tareas de reconocimiento de imágenes, donde se usan kernels que se aprenden mediante la técnica de back-propagation al entrenar la red, mientras que las ondículas son fijas y no se aprenden.

Por otro lado, fuera del contexto de las redes neuronales convolucionales, cabe destacar que esta modelización se puede relacionar también con algoritmos clásicos de Visión por Computador, como SIFT [Low04], el cual se emplea para identificar puntos de interés en imágenes. De esta forma, usando las ondículas adecuadas (aquellas cuya forma y escala capturen efectivamente las variaciones locales de intensidad y orientación en las imágenes, como las ondículas de Gabor [Wik24a], las cuales detallaremos más adelante), los coeficientes de primer orden $S[\lambda_1]f$ serían equivalentes a los coeficientes obtenidos con este algoritmo.

2.3.3. No-expansividad en la distancia entre funciones

El propagador $U_J f = \{A_J f, (|W[\lambda]f|)_{\lambda \in \Lambda_J}\}$ es no-expansivo, ya que la transformada de ondículas W_J es unitaria según la [Proposición 2.1](#), y el módulo es no-expansivo en el sentido de que $| |a| - |b| | \leq |a - b|$ para cualquier $(a, b) \in \mathbb{C}^2$. Esto se cumple tanto si f es una función real o como si es una función compleja. En consecuencia:

$$\|U_J f - U_J h\|^2 = \|A_J f - A_J h\|^2 + \sum_{\lambda \in \Lambda_J} \| |W[\lambda]f| - |W[\lambda]h| \|^2 \leq \|W_J f - W_J h\|^2 \leq \|f - h\|^2.$$

Dado que W_J es unitaria, si tomamos $h = 0$, se verifica que $\|U_J f\| = \|f\|$, lo que confirma que el operador U_J preserva la norma.

Para cualquier conjunto de caminos Ω , las normas de $S_J[\Omega]f$ y $U[\Omega]f$ son:

$$\|S_J[\Omega]f\|^2 = \sum_{p \in \Omega} \|S_J[p]f\|^2 \quad y \quad \|U[\Omega]f\|^2 = \sum_{p \in \Omega} \|U[p]f\|^2.$$

2.3. Propagación de la dispersión y conservación de la norma

Puesto que $S_J[\mathcal{P}_J]$ itera en U_J , que es no-expansivo, la siguiente proposición muestra que $S_J[\Omega]f$ también es no-expansivo.

Proposición 2.3. *La transformada de dispersión de ventana es no-expansiva:*

$$\forall(f, h) \in L^2(\mathbb{R}^d)^2, \quad \|S_J[\mathcal{P}_J]f - S_J[\mathcal{P}_J]h\| \leq \|f - h\|.$$

Demostración. Dado que U_J es no-expansivo, partiendo de la ecuación (2.7) que indica:

$$U_J(U[\Lambda_J^m]f) = \{S_J[\Lambda_J^m]f, (U[\Lambda_J^{m+1}]f)_{\lambda \in \Lambda_J}\},$$

tenemos que:

$$\begin{aligned} \|U[\Lambda_J^m]f - U[\Lambda_J^m]h\|^2 &\geq \|U_J(U[\Lambda_J^m]f) - U_J(U[\Lambda_J^m]h)\|^2 \\ &= \|S_J[\Lambda_J^m]f - S_J[\Lambda_J^m]h\|^2 + \|U[\Lambda_J^{m+1}]f - U[\Lambda_J^{m+1}]h\|^2. \end{aligned}$$

Al sumar en m cuando m tiende a infinito, obtenemos:

$$\sum_{m=0}^{\infty} \|U[\Lambda_J^m]f - U[\Lambda_J^m]h\|^2 \geq \sum_{m=0}^{\infty} \|S_J[\Lambda_J^m]f - S_J[\Lambda_J^m]h\|^2 + \sum_{m=0}^{\infty} \|U[\Lambda_J^{m+1}]f - U[\Lambda_J^{m+1}]h\|^2,$$

que equivale a:

$$\sum_{m=0}^{\infty} \|U[\Lambda_J^m]f - U[\Lambda_J^m]h\|^2 - \sum_{m=0}^{\infty} \|U[\Lambda_J^{m+1}]f - U[\Lambda_J^{m+1}]h\|^2 \geq \sum_{m=0}^{\infty} \|S_J[\Lambda_J^m]f - S_J[\Lambda_J^m]h\|^2.$$

Si ahora nos fijamos en el lado izquierdo de la desigualdad, se cancelan todos los términos salvo $m = 0$, y, como $\Lambda_J^0 = \emptyset$, se tiene que:

$$\|U[\Lambda_J^0]f - U[\Lambda_J^0]h\|^2 = \|U[\emptyset]f - U[\emptyset]h\|^2 = \|f - h\|^2.$$

Por otro lado, para el miembro derecho de la desigualdad anterior, obtenemos que:

$$\sum_{m=0}^{\infty} \|S_J[\Lambda_J^m]f - S_J[\Lambda_J^m]h\|^2 = \|S_J[\mathcal{P}_J]f - S_J[\mathcal{P}_J]h\|^2.$$

2. Teoría de Mallat sobre Redes de Dispersión

Finalmente, se ha demostrado que:

$$\|S_J[\mathcal{P}_J]f - S_J[\mathcal{P}_J]h\|^2 \leq \|f - h\|^2,$$

y, por lo tanto, que la transformada de dispersión de ventana es no-expansiva.

□

2.3.4. Conservación de la norma

En la [Subsección 2.2.3](#) se obtuvo que cada coeficiente $U[\lambda]f = |f * \psi_\lambda|$ capturaba la energía de f en una banda de frecuencia cubierta por $\widehat{\psi}_\lambda$, propagando dicha energía hacia frecuencias menores. El siguiente resultado prueba que toda la energía del propagador de dispersión alcanza la frecuencia mínima 2^{-J} y es atrapada por el filtro de paso bajo ϕ_{2^J} . Como la energía se approxima a 0 al aumentar la longitud del camino, el teorema que demostraríamos implica que $\|S_J[\mathcal{P}_J]f\| = \|f\|$. Este resultado también es válido para funciones complejas al incorporar caminos negativos.

Para la demostración de la conservación de la norma de la transformada de dispersión de ventana necesitamos unos resultados previos:

Lema 2.3. *Sea h una función tal que $h \geq 0$. Entonces, para todo $f \in L^2(\mathbb{R}^d)$, se cumple que:*

$$(|f * \psi_\lambda| * h) \geq \sup_{\eta \in \mathbb{R}^d} |f * \psi_\lambda * h_\eta| \quad \text{con} \quad h_\eta = h(x)e^{i\langle x, \eta \rangle}.$$

Demostración.

$$\begin{aligned} (|f * \psi_\lambda| * h)(x) &= \int \left| \int f(v)\psi_\lambda(u-v)dv \right| h(x-u)du \\ &= \int \left| \int f(v)\psi_\lambda(u-v)e^{i\langle(x-u), \eta\rangle}h(x-u)dv \right| du \\ &\geq \left| \int \int f(v)\psi_\lambda(u-v)e^{i\langle(x-u), \eta\rangle}h(x-u)dudv \right| \\ &= \left| \int f(v) \int \psi_\lambda(x-v-u')h(u')e^{i\langle u', \eta \rangle}du'dv \right| \\ &= \left| \int f(v)(\psi_\lambda * h_\eta)(x-v)dv \right| \\ &= |f * \psi_\lambda * h_\eta|, \end{aligned}$$

donde se ha usado el cambio de variable $u' = x - u$, con Jacobiano igual a uno.

□

A continuación definimos el concepto de *ondícula admisible*.

Definición 2.11. Una ondícula de dispersión ψ se considera *admisible* si existe un valor $\eta \in \mathbb{R}^d$ y una función $\rho \geq 0$, con $|\widehat{\rho}(\omega)| \leq |\widehat{\phi}(2\omega)|$ y $\widehat{\rho}(0) = 1$, tal que la función

$$\widehat{\Psi}(\omega) = |\widehat{\rho}(\omega - \eta)|^2 - \sum_{k=1}^{\infty} k \left(1 - \left| \widehat{\rho}(2^{-k}(\omega - \eta)) \right|^2 \right),$$

satisface:

$$\alpha = \inf_{1 \leq |\omega| \leq 2} \sum_{j=-\infty}^{\infty} \sum_{r \in G} \widehat{\Psi}(2^{-j}r^{-1}\omega) \left| \widehat{\psi}(2^{-j}r^{-1}\omega) \right|^2 > 0. \quad (2.8)$$

Con esta definición en mente podemos comprobar que se da el siguiente lema que demuestra que el propagador dispersa la energía progresivamente hacia bajas frecuencias.

Lema 2.4. Si se satisface (2.8) y:

$$\|f\|_w^2 = \sum_{j=0}^{\infty} \sum_{r \in G^+} j \|W[2^j r]f\|^2 < \infty,$$

entonces:

$$\frac{\alpha}{2} \|U[\mathcal{P}_J]f\|^2 \leq \max(J+1, 1) \|f\|^2 + \|f\|_w^2. \quad (2.9)$$

La demostración de este lema se puede encontrar en ([Mal12], Apéndice A). No ha sido incluida en este trabajo debido a su elevada complejidad técnica.

Con estos resultados, estamos listos para enunciar el teorema principal de esta sección, el cual nos proporcionará la conservación de la norma del operador de dispersión de ventana.

Teorema 2.2. Si la ondícula es admisible, entonces, para todo $f \in L^2(\mathbb{R}^d)$, se cumple que:

$$\lim_{m \rightarrow \infty} \|U[\Lambda_J^m]f\|^2 = \lim_{m \rightarrow \infty} \sum_{n=m}^{\infty} \|S_J[\Lambda_J^n]f\|^2 = 0,$$

y:

$$\|S_J[\mathcal{P}_J]f\| = \|f\|.$$

Demostración. Esta prueba se divide en dos partes, la primera de las cuales consistirá en demostrar que la condición (2.9) implica que $\lim_{m \rightarrow \infty} \|U[\Lambda_J^m]f\|^2 = 0$.

La clave radica en el **Lema 2.3**, que proporciona una cota inferior para $|f * \psi_\lambda|$ convolucionada con una función positiva. Como:

2. Teoría de Mallat sobre Redes de Dispersión

$$\|U[\mathcal{P}_J]f\|^2 = \sum_{m=0}^{\infty} \|U[\Lambda_J^m]f\|^2,$$

si $\|f\|_w < \infty$ entonces (2.9) implica que $\lim_{m \rightarrow \infty} \|U[\Lambda_J^m]f\| = 0$. Este resultado se extiende a $L^2(\mathbb{R}^d)$ por densidad. Como $\phi \in L^1(\mathbb{R}^d)$ y $\widehat{\phi}(0) = 1$, cualquier $f \in L^2(\mathbb{R}^d)$ satisface $\lim_{n \rightarrow -\infty} \|f - f_n\| = 0$, donde $f_n = f * \phi_{2^n}$ y $\phi_{2^n} = 2^{-n}\phi(2^{-n}x)$. Se demuestra por tanto que $\lim_{m \rightarrow \infty} \|U[\Lambda_J^m]f_n\| = 0$ viendo que $\|f_n\|_w < \infty$. De hecho:

$$\|W[2^j r]f_n\|^2 = \int (|\widehat{f}(\omega)|^2 |\widehat{\phi}(2^n \omega)|^2 |\widehat{\psi}(2^{-j}r^{-1}\omega)|^2) d\omega \leq C 2^{-2n-2j} \int |\widehat{f}(\omega)|^2 d\omega,$$

porque llega un punto donde ψ desaparece, entonces $|\widehat{\psi}(\omega)| = O(|\omega|)$, y las derivadas de ϕ están en $L^1(\mathbb{R}^d)$, lo que implica que $|\omega| |\widehat{\phi}(\omega)|$ está acotado, luego, $\|f_n\|_w < \infty$.

Como $U[\Lambda^m]$ es no-expansivo, se tiene $\|U[\Lambda_J^m]f - U[\Lambda_J^m]f_n\| \leq \|f - f_n\|$, por lo que:

$$\|U[\Lambda_J^m]f\| \leq \|f - f_n\| + \|U[\Lambda_J^m]f_n\|.$$

Ya que $\lim_{n \rightarrow -\infty} \|f - f_n\| = 0$ y $\lim_{m \rightarrow \infty} \|U[\Lambda_J^m]f_n\| = 0$, tenemos que:

$$\lim_{m \rightarrow \infty} \|U[\Lambda_J^m]f\|^2 = 0,$$

para todo $f \in L^2(\mathbb{R}^d)$.

La segunda parte de esta prueba consiste en demostrar que las siguientes expresiones son equivalentes:

$$\lim_{m \rightarrow \infty} \|U[\Lambda_J^m]f\|^2 = 0 \iff \lim_{m \rightarrow \infty} \sum_{n=m}^{\infty} \|S_J[\Lambda_J^n]f\|^2 = 0 \iff \|S_J[\mathcal{P}_J]f\|^2 = \|f\|^2.$$

Primero probamos que:

$$\lim_{m \rightarrow \infty} \|U[\Lambda_J^m]f\|^2 = 0 \iff \lim_{m \rightarrow \infty} \sum_{n=m}^{\infty} \|S_J[\Lambda_J^n]f\|^2 = 0.$$

Como $\|U_J h\| = \|h\|$, para todo $h \in L^2(\mathbb{R}^d)$, y $U_J(U[\Lambda_J^n]f) = \{S_J[\Lambda_J^n]f, U[\Lambda_J^{n+1}]f\}$, se tiene que:

2.3. Propagación de la dispersión y conservación de la norma

$$\|U[\Lambda_J^n]f\|^2 = \|U_J(U[\Lambda_J^n]f)\|^2 = \|S_J[\Lambda_J^n]f\|^2 + \|U[\Lambda_J^{n+1}]f\|^2. \quad (2.10)$$

Sumando en $m \leq n < \infty$, se obtiene:

$$\begin{aligned} \sum_{n=m}^{\infty} \|U[\Lambda_J^n]f\|^2 &= \sum_{n=m}^{\infty} \|S_J[\Lambda_J^n]f\|^2 + \sum_{n=m}^{\infty} \|U[\Lambda_J^{n+1}]f\|^2 \\ \iff \sum_{n=m}^{\infty} \|U[\Lambda_J^n]f\|^2 - \sum_{n=m}^{\infty} \|U[\Lambda_J^{n+1}]f\|^2 &= \sum_{n=m}^{\infty} \|S_J[\Lambda_J^n]f\|^2. \end{aligned}$$

En el lado izquierdo, los términos se cancelan salvo cuando $n = m$, lo que nos lleva a:

$$\|U[\Lambda_J^m]f\|^2 = \sum_{n=m}^{\infty} \|S_J[\Lambda_J^n]f\|^2.$$

Tomando límites cuando $m \rightarrow \infty$, obtenemos:

$$\lim_{m \rightarrow \infty} \|U[\Lambda_J^m]f\|^2 = \lim_{m \rightarrow \infty} \sum_{n=m}^{\infty} \|S_J[\Lambda_J^n]f\|^2.$$

Por otro lado, sumando en la ecuación (2.10) para $0 \leq n < m$, se obtiene:

$$\begin{aligned} \sum_{n=0}^{m-1} \|U[\Lambda_J^n]f\|^2 &= \sum_{n=0}^{m-1} \|S_J[\Lambda_J^n]f\|^2 + \sum_{n=0}^{m-1} \|U[\Lambda_J^{n+1}]f\|^2 \\ \iff \sum_{n=0}^{m-1} \|U[\Lambda_J^n]f\|^2 - \sum_{n=0}^{m-1} \|U[\Lambda_J^{n+1}]f\|^2 &= \sum_{n=0}^{m-1} \|S_J[\Lambda_J^n]f\|^2. \end{aligned}$$

Como los términos del lado izquierdo se cancelan salvo cuando $n = 0$, considerando que $f = U[\Lambda_J^0]f$, llegamos a:

$$\|f\|^2 = \|U[\Lambda_J^m]f\|^2 + \sum_{n=0}^{m-1} \|S_J[\Lambda_J^n]f\|^2.$$

Si ahora tomamos el límite cuando $m \rightarrow \infty$ obtenemos:

$$\lim_{m \rightarrow \infty} \|f\|^2 = \lim_{m \rightarrow \infty} \|U[\Lambda_J^m]f\|^2 + \lim_{m \rightarrow \infty} \sum_{n=0}^{m-1} \|S_J[\Lambda_J^n]f\|^2$$

2. Teoría de Mallat sobre Redes de Dispersión

$$\begin{aligned} \iff \|f\|^2 &= \lim_{m \rightarrow \infty} \|U[\Lambda_J^m]f\|^2 + \sum_{n=0}^{\infty} \|S_J[\Lambda_J^n]f\|^2 \\ \iff \|f\|^2 &= \lim_{m \rightarrow \infty} \|U[\Lambda_J^m]f\|^2 + \|S_J[\mathcal{P}_J]f\|^2. \end{aligned}$$

De manera que se puede observar claramente que:

$$\|f\|^2 = \|S_J[\mathcal{P}_J]f\|^2 \iff \lim_{m \rightarrow \infty} \|U[\Lambda_J^m]f\|^2 = 0.$$

Con lo que queda probado el teorema. □

Esta demostración muestra que el propagador dispersa gradualmente la energía hacia frecuencias más bajas. La energía de $U[p]f$ se concentra principalmente en los caminos de frecuencia decreciente $p = (\lambda_k)_{k \leq m}$, donde $|\lambda_{k+1}| < |\lambda_k|$.

El decaimiento de $\sum_{n=m}^{\infty} \|S_J[\Lambda_J^n]f\|^2$ sugiere que podemos descartar caminos de longitud superior a un cierto $m > 0$. En problemas de clasificación, se suele limitar el camino a $m = 3$ debido al decaimiento exponencial de $\|S_J[\Lambda_J^n]f\|^2$, por ejemplo, en tareas de tratamiento de imágenes y audio.

Este último teorema requiere además de una transformada de ondículas unitaria y admisible que cumpla con la condición de Littlewood-Paley $\beta \sum_{(j,r) \in \mathbb{Z} \times G} |\widehat{\psi}(2^j r \omega)|^2 = 1$.

También se necesita la existencia de una función $\rho \geq 0$ y un parámetro $\eta \in \mathbb{R}^d$, con $|\widehat{\rho}(\omega)| \leq |\widehat{\phi}(2\omega)|$, tal que:

$$\sum_{(j,r) \in \mathbb{Z} \times G} |\widehat{\psi}(2^j r \omega)|^2 |\widehat{\rho}(2^j r \omega - \eta)|^2,$$

sea lo suficientemente grande para garantizar que $\alpha > 0$. Esto se puede deducir de lo explicado a partir de la [Definición 2.3](#) de la [Subsección 2.1.2](#), con $\psi(x) = e^{i\langle x, \eta \rangle} \Theta(x)$ y $\widehat{\psi} = \widehat{\Theta}(\omega - \eta)$, donde tanto $\widehat{\Theta}$ como $\widehat{\rho}$ tienen su energía concentrada en los mismos dominios de frecuencias bajas.

Existen ondículas que verifican las exigencias establecidas en el teorema anterior. Una clase común que satisface estas condiciones son las ondículas de Gabor [[Wik24a](#)], ya mencionadas anteriormente. La ecuación de una ondícula de Gabor unidimensional es una función gaussiana modulada por una exponencial compleja, expresada de la siguiente manera:

$$\psi(x) = e^{-\frac{(x-x_0)^2}{\sigma^2}} e^{-i\omega_0(x-x_0)},$$

donde x_0 es la posición central de la ondícula, $\sigma > 0$ controla la extensión espacial (ancho de la gaussiana), y ω_0 es la frecuencia de modulación. Esta definición resalta que las ondículas de Gabor están localizadas tanto en tiempo como en frecuencia, ya que el término gaussiano asegura una caída exponencial a medida que x se aleja de x_0 .

También vale la pena señalar que transformada de Fourier de una ondícula de Gabor también es una ondícula de Gabor y está dada por:

$$\widehat{\psi}(\omega) = \sigma e^{-(\omega-\omega_0)^2\sigma^2} e^{-ix_0(\omega-\omega_0)},$$

lo que implica que $\widehat{\psi}(\omega)$ está centrada en $\omega = \omega_0$ y tiene un soporte en frecuencias bajas, controlado por σ .

Este tipo de ondículas cumple con las condiciones de Littlewood-Paley al generar una base ortonormal mediante rotaciones y escalas adecuadas, asegurando la cobertura del dominio de frecuencias de manera eficiente. Además, la función ρ requerida para garantizar que $\alpha > 0$ puede definirse como una gaussiana de baja frecuencia.

2.4. Invarianza horizontal por traslaciones

2.4.1. No-expansividad en conjuntos de caminos

Vamos a demostrar en primer lugar que $\|S_J[\mathcal{P}_J]f - S_J[\mathcal{P}_J]h\|$ es no-expansivo cuando incrementa J , y que converge cuando $J \rightarrow \infty$. Esto establece una distancia límite que, como veremos a continuación, es invariante por traslaciones.

Vamos a necesitar el siguiente lema:

Lema 2.5. *Para las ondículas que cumplen con la propiedad mencionada en la [Proposición 2.1](#), y para toda función real $f \in L^2(\mathbb{R}^d)$ y todo $q \in \mathbb{Z}$, se tiene que:*

$$\|f * \phi_{2^J}\|^2 + \sum_{-q \geq l > -J} \sum_{r \in G^+} \|f * \psi_{2^J r}\|^2 = \|f * \phi_{2^0}\|^2.$$

Demostración. En primer lugar, vamos a ver que de la [Proposición 2.1](#) se deduce la siguiente relación:

$$|\widehat{\phi}(2^J \omega)|^2 + \sum_{-q \geq l > -J} \sum_{r \in G^+} |\widehat{\psi}(2^{-r} \widehat{\omega})|^2 = |\widehat{\phi}(2^J \omega)|^2.$$

Para ello, de la expresión:

2. Teoría de Mallat sobre Redes de Dispersión

$$\frac{1}{2} \sum_{j=-\infty}^{\infty} \sum_{r \in G} |\widehat{\psi}(2^{-j}r^{-1}\omega)|^2 = 1 \quad \text{y} \quad |\widehat{\phi}(\omega)|^2 = \frac{1}{2} \sum_{j=-\infty}^0 \sum_{r \in G} |\widehat{\psi}(2^{-j}r^{-1}\omega)|^2,$$

tenemos de manera análoga a cómo vimos en la demostración de la [Proposición 2.1](#) que:

$$\forall J \in \mathbb{Z}, \quad |\widehat{\phi}(2^J\omega)|^2 + \frac{1}{2} \sum_{j>-J, r \in G} |\widehat{\psi}(2^{-j}r^{-1}\omega)|^2 = 1.$$

Y partiendo del sumatorio obtenemos lo siguiente:

$$|\widehat{\phi}(2^J\omega)|^2 + \frac{1}{2} \sum_{-q \geq j > -J, r \in G} |\widehat{\psi}(2^{-j}r^{-1}\omega)|^2 = \frac{1}{2} \sum_{j > -q, r \in G} |\widehat{\psi}(2^{-j}r^{-1}\omega)|^2 = |\widehat{\phi}(2^q\omega)|^2.$$

Ahora multiplicamos esta expresión por $|\widehat{f}(\omega)|^2$, obteniendo:

$$|\widehat{f}(\omega)|^2 |\widehat{\phi}(2^J\omega)|^2 + \frac{1}{2} \sum_{-q \geq j > -J, r \in G} |\widehat{f}(\omega)|^2 |\widehat{\psi}(2^{-j}r^{-1}\omega)|^2 = |\widehat{f}(\omega)|^2 |\widehat{\phi}(2^q\omega)|^2.$$

Integrando en ω :

$$\begin{aligned} \int |\widehat{f}(\omega)|^2 |\widehat{\phi}(2^J\omega)|^2 d\omega + \frac{1}{2} \sum_{-q \geq j > -J, r \in G} \int |\widehat{f}(\omega)|^2 |\widehat{\psi}(2^{-j}r^{-1}\omega)|^2 d\omega \\ = \int |\widehat{f}(\omega)|^2 |\widehat{\phi}(2^q\omega)|^2 d\omega. \end{aligned}$$

Ahora estamos en condiciones de aplicar el [Teorema 2.1](#), lo que nos da que la expresión anterior es equivalente a:

$$\int |(f * \phi_{2^J})(x)|^2 dx + \sum_{-q \geq j > -J, r \in G} \int |(f * \psi_{2^j r})(x)|^2 dx = \int |(f * \phi_{2^q})(x)|^2 dx.$$

Al tener en cuenta que f es una función real, podemos afirmar que $\|f * \psi_{2^j r}\| = \|f * \psi_{2^{j-r}}\|$. Junto con la definición de norma en $L^2(\mathbb{R}^d)$, se tiene que:

$$\|f * \phi_{2^J}\|^2 + \sum_{-q \geq l > -J} \sum_{r \in G^+} \|f * \psi_{2^l r}\|^2 = \|f * \phi_{2^q}\|^2.$$

□

Proposición 2.4. Para todo $f, h \in L^2(\mathbb{R}^d)$ y $J \in \mathbb{Z}$ se cumple que:

$$\|S_{J+1}[\mathcal{P}_{J+1}]f - S_{J+1}[\mathcal{P}_{J+1}]h\| \leq \|S_J[\mathcal{P}_J]f - S_J[\mathcal{P}_J]h\|. \quad (2.11)$$

Demostración. En primer lugar, vamos a transformar la condición que queremos demostrar en (2.11) a otra equivalente y que será más fácil de probar.

Si recordamos la definición de \mathcal{P}_J , este es un conjunto de caminos finitos $p = (\lambda_1, \dots, \lambda_m)$ tal que $\lambda_k \in \Lambda_J$ y $|\lambda_k| = 2^k > 2^{-J}$. Luego, todo camino $p' \in \mathcal{P}_{J+1}$ puede ser únicamente escrito como una extensión de un camino $p \in \mathcal{P}_J$, donde p es el prefijo más grande de p' que pertenece a \mathcal{P}_J , y $p' = p + q$ para algún $q \in \mathcal{P}_{J+1}$. De hecho, podemos definir el conjunto de todas las extensiones de $p \in \mathcal{P}_J$ en \mathcal{P}_{J+1} como:

$$\mathcal{P}_{J+1}^p = \{p\} \cup \{p + 2^{-J}r + p''\}_{r \in G^+, p'' \in \mathcal{P}_{J+1}}.$$

Esto define una partición disjunta de $\mathcal{P}_{J+1} = \bigcup_{p \in \mathcal{P}_J} \mathcal{P}_{J+1}^p$. Debemos probar que dichas extensiones son no-expansivas, es decir:

$$\sum_{p' \in \mathcal{P}_{J+1}^p} \|S_{J+1}[p']f - S_{J+1}[p']h\|^2 \leq \|S_J[p]f - S_J[p]h\|^2. \quad (2.12)$$

Observamos que la condición (2.12) es equivalente a (2.11) al sumar sobre todo $p \in \mathcal{P}_J$. Por lo tanto, al demostrar (2.12), obtendremos el resultado que buscamos.

Usando el [Lema 2.5](#) con la función $g = U[p]f - U[p]h$, se tiene que:

$$\|g * \phi_{2^J}\|^2 = \|g * \phi_{2^{J+1}}\|^2 + \sum_{r \in G^+} \|g * \psi_{2^{-J}r}\|^2.$$

Así, sustituyendo el valor de g por el que hemos definido antes y aplicando la propiedad distributiva de la convolución, obtenemos:

$$\begin{aligned} & \| (U[p]f * \phi_{2^J}) - (U[p]h * \phi_{2^J}) \|^2 \\ &= \| (U[p]f * \phi_{2^{J+1}}) - (U[p]h * \phi_{2^{J+1}}) \|^2 + \sum_{r \in G^+} \| (U[p]f * \psi_{2^{-J}r}) - (U[p]h * \psi_{2^{-J}r}) \|^2. \end{aligned}$$

Como $U[p]f * \phi_{2^J} = S_J[p]f$, para todo $f \in L^2(\mathbb{R}^d)$, esta última expresión equivale a:

$$\|S_J[p]f - S_J[p]h\|^2 = \|S_{J+1}[p]f - S_{J+1}[p]h\|^2 + \sum_{r \in G^+} \| (U[p]f * \psi_{2^{-J}r}) - (U[p]h * \psi_{2^{-J}r}) \|^2.$$

2. Teoría de Mallat sobre Redes de Dispersión

Aplicando que $|U[p]f * \psi_{2^{-J}r}| = U[p + 2^{-J}r]f$ y la propiedad de la norma, que establece que $\|g - h\| \geq \| |g| \| - \| |h| \|$, se obtiene:

$$\|S_J[p]f - S_J[p]h\|^2 \geq \|S_{J+1}[p]f - S_{J+1}[p]h\|^2 + \sum_{r \in G^+} \|U[p + 2^{-J}r]f - U[p + 2^{-J}r]h\|^2.$$

Como $S_{J+1}[\mathcal{P}_{J+1}](U[p + 2^{-J}r]f) = \{S_{J+1}[p + 2^{-J}r + p'']\}_{p'' \in \mathcal{P}_{J+1}}$ y sabemos que $S_{J+1}[\mathcal{P}_{J+1}]f$ es no-expansivo por la [Proposición 2.3](#), podemos aplicar la desigualdad anterior para obtener:

$$\begin{aligned} & \|S_J[p]f - S_J[p]h\|^2 \\ & \geq \|S_{J+1}[p]f - S_{J+1}[p]h\|^2 + \sum_{p'' \in \mathcal{P}_{J+1}} \sum_{r \in G^+} \|S_{J+1}[p + 2^{-J}r + p'']f - S_{J+1}[p + 2^{-J}r + p'']h\|^2. \end{aligned}$$

En particular:

$$\|S_J[p]f - S_J[p]h\|^2 \geq \sum_{p'' \in \mathcal{P}_{J+1}} \sum_{r \in G^+} \|S_{J+1}[p + 2^{-J}r + p'']f - S_{J+1}[p + 2^{-J}r + p'']h\|^2,$$

lo que prueba [\(2.12\)](#). □

2.4.2. Demostración de la invarianza por traslaciones

La proposición anterior nos demuestra que $\|S_J[\mathcal{P}_J]f - S_J[\mathcal{P}_J]h\|$ es positivo y no creciente cuando J aumenta, y converge. Dado que $S_J[\mathcal{P}_J]$ es no-expansivo, su límite también lo será:

$$\forall f, h \in L^2(\mathbb{R}^d), \quad \lim_{J \rightarrow \infty} \|S_J[\mathcal{P}_J]f - S_J[\mathcal{P}_J]h\| \leq \|f - h\|.$$

Para ondículas de dispersión admisibles que satisfacen la condición [\(2.8\)](#), el [Teorema 2.2](#) asegura que si $\|S_J[\mathcal{P}_J]f\| = \|f\|$, entonces $\lim_{J \rightarrow \infty} \|S_J[\mathcal{P}_J]f\| = \|f\|$.

En el siguiente teorema que enunciemos, se demostrará que el límite es invariante por traslaciones, pero antes necesitamos dos resultados previos:

Lema 2.6. *Sea $K : L^2(\mathbb{R}^d) \rightarrow L^2(\mathbb{R}^d)$ un operador tal que, $\forall f \in L^2(\mathbb{R}^d)$, $(Kf)(x) = \int f(u)k(x, u)du$ verificando que:*

$$\int |k(x, u)| dx \leq C,$$

y:

$$\int |k(x, u)| du \leq C,$$

con $C > 0$. Entonces $\|K\| \leq C$, donde $\|K\|$ es la norma en $L^2(\mathbb{R}^d)$ de K .

Este lema es conocido como el *lema de Schur*. En este caso, se presenta como un lema auxiliar que será utilizado para la demostración del siguiente resultado, por lo que no incluiremos su prueba.

Lema 2.7. Existe una constante $C \in \mathbb{R}$ tal que para todo $\tau \in C^2(\mathbb{R}^d)$ con $\|\text{Jac}(\tau)\|_\infty \leq \frac{1}{2}$ se cumple que:

$$\|T_\tau(A_J f) - A_J f\| \leq C \|f\| 2^{-J} \|\tau\|_\infty.$$

Demostración. La normal del operador $k_J = T_\tau A_J - A_J$ se calcula aplicando el [Lema 2.6](#) a su kernel:

$$k_J(x, u) = \phi_{2^J}(x - \tau(x) - u) - \phi_{2^J}(x - u).$$

Si observamos la expresión anterior, cuando $x = 0 = u$, tenemos que:

$$k_J(0, 0) = \phi_{2^J}(0) - \phi_{2^J}(0) = 0.$$

Si ahora calculamos su polinomio de Taylor de primer orden centrado en el $(0, 0)$, se obtiene:

$$k_J(x, u) = k_J(0, 0) + \int_0^1 (\text{Jac}(\phi_{2^J}))(x - t\tau(x) - u) \tau(x) dt.$$

Si ahora calculamos el módulo obtenemos que:

2. Teoría de Mallat sobre Redes de Dispersión

$$\begin{aligned}
|k_J(x, u)| &= \left| k_J(0, 0) + \int_0^1 (\text{Jac}(\phi_{2J}))(x - t\tau(x) - u)\tau(x)dt \right| \\
&\leq |k_J(0, 0)| + \left| \int_0^1 (\text{Jac}(\phi_{2J}))(x - t\tau(x) - u)\tau(x)dt \right| \\
&\leq \left| \int_0^1 (\text{Jac}(\phi_{2J}))(x - t\tau(x) - u)\tau(x)dt \right| \\
&\leq \int_0^1 |(\text{Jac}(\phi_{2J}))(x - t\tau(x) - u)\tau(x)|dt \\
&= |\tau(x)| \int_0^1 |(\text{Jac}(\phi_{2J}))(x - t\tau(x) - u)|dt \\
&\leq \|\tau\|_\infty \int_0^1 |(\text{Jac}(\phi_{2J}))(x - t\tau(x) - u)|dt.
\end{aligned}$$

Si ahora integramos en u y aplicamos el conocido *teorema de Fubini* para intercambiar las integrales del lado derecho de la desigualdad obtenemos:

$$\begin{aligned}
\int |k_J(x, u)|du &\leq \|\tau\|_\infty \int \int_0^1 |(\text{Jac}(\phi_{2J}))(x - t\tau(x) - u)|dtdu \\
&= \|\tau\|_\infty \int_0^1 \int |(\text{Jac}(\phi_{2J}))(x - t\tau(x) - u)|dudt.
\end{aligned}$$

Por otro lado, vamos a comprobar que:

$$((\text{Jac}(\phi_{2J}))(x)) = 2^{-2J}(\text{Jac}(\phi))(2^{-J}x).$$

Para esto, recordemos que $\phi_{2J}(x) = 2^{-J}\phi(2^{-J}x)$, por lo que:

$$(\text{Jac}(\phi_{2J}))(x) = \text{Jac}((2^{-J}\phi(2^{-J}x))) = 2^{-2J}(\text{Jac}(\phi))(2^{-J}x).$$

De esta manera, al realizar un cambio de variable $u' = x - t\tau(x) - u$, se obtiene:

$$\begin{aligned}
\int |k_J(x, u)|du &\leq \|\tau\|_\infty 2^{-2J} \int |(\text{Jac}(\phi))(2^{-J}u')| du' \\
&= 2^{-J} \|\tau\|_\infty \|\text{Jac}(\phi)\|_1.
\end{aligned}$$

2.4. Invarianza horizontal por traslaciones

Si ahora realizamos un procedimiento análogo, pero integrando en x en lugar de u , obtenemos:

$$\int |k_J(x, u)| dx \leq \|\tau\|_\infty \int_0^1 \int |(Jac(\phi_{2^J}))(x - t\tau(x) - u)| dx dt.$$

Aplicamos ahora un cambio de variable $v = x - t\tau(x)$ y determinamos su Jacobiano:

$$Jac(v) = Jac(x - t\tau(x)) = Jac(x) - Jac(t\tau(x)) = \text{Id} - tJac(\tau(x)) = \text{Id} - t(Jac(\tau))(x).$$

Por hipótesis, $\|Jac(\tau)\|_\infty \leq \frac{1}{2}$, por lo que podemos establecer una cota para el determinante del Jacobiano:

$$\det(Jac(v)) = (1 - t(Jac(\tau))(x))^d \geq (1 - \|Jac(\tau)\|_\infty)^d \geq 2^{-d}.$$

Aplicando ahora el cambio de variable a la integral:

$$\begin{aligned} \int |k_J(x, u)| dx &\leq \|\tau\|_\infty 2^d \int_0^1 \int |(Jac(\phi_{2^J}))(v - u)| dv dt \\ &= 2^{-J} \|\tau\|_\infty \|Jac(\phi)\|_1 2^d \\ &= 2^{-J+d} \|\tau\|_\infty \|Jac(\phi)\|_1. \end{aligned}$$

Entre las cotas superiores obtenidas, esta última es la mayor, por lo que aplicamos el [Lema 2.6](#) y concluimos la demostración:

$$\|T_\tau A_J - A_J\| \leq 2^{-J+d} \|\tau\|_\infty \|Jac(\phi)\|_1.$$

□

Con esto, ya contamos con las herramientas necesarias para presentar y demostrar el resultado principal de esta sección, que asegura que el operador de dispersión de ventana que estamos desarrollando, y que modeliza una red neuronal convolucional, es invariante por traslaciones.

Teorema 2.3. *Para ondículas de dispersión admisibles, se verifica que:*

$$\forall f \in L^2(\mathbb{R}^d), \forall t \in \mathbb{R}^d, \quad \lim_{J \rightarrow \infty} \|S_J[\mathcal{P}_J]f - S_J[\mathcal{P}_J](T_tf)\| = 0.$$

2. Teoría de Mallat sobre Redes de Dispersión

Demostración. Fijamos $f \in L^2(\mathbb{R}^d)$. Teniendo en cuenta la conmutatividad $S_J[\mathcal{P}_J](T_t f) = T_t(S_J[\mathcal{P}_J]f)$ y usando la definición $S_J[\mathcal{P}_J]f = A_J(U[\mathcal{P}_J]f)$, obtenemos:

$$\|S_J[\mathcal{P}_J](T_t f) - S_J[\mathcal{P}_J]f\| = \|T_t(A_J(U[\mathcal{P}_J]f)) - A_J(U[\mathcal{P}_J]f)\| \leq \|T_t A_J - A_J\| \|U[\mathcal{P}_J]f\|.$$

Aplicando el [Lema 2.7](#) con $\tau = t$, tenemos que $\|\tau\|_\infty = |t|$, y así:

$$\|T_t A_J - A_J\| \leq C 2^{-J} |t|,$$

con $C \in \mathbb{R}$. Usando esta última desigualdad, obtenemos:

$$\|S_J[\mathcal{P}_J](T_t f) - S_J[\mathcal{P}_J]f\| \leq \|T_t A_J - A_J\| \|U[\mathcal{P}_J]f\| \leq C 2^{-J} |t| \|U[\mathcal{P}_J]f\|.$$

Dado que se cumple la condición de admisibilidad para ondículas (2.8), según (2.9) en el [Lema 2.4](#), para $J > 1$ tenemos que:

$$\frac{\alpha}{2} \|U[\mathcal{P}_J]f\|^2 \leq (J+1) \|f\|^2 + \|f\|_w^2.$$

Y de esta forma, podemos obtener una cota superior para la norma de $\|U[\mathcal{P}_J]f\|$:

$$\|U[\mathcal{P}_J]f\|^2 \leq ((J+1) \|f\|^2 + \|f\|_w^2) 2\alpha^{-1}.$$

Si $\|f\|_w < \infty$, entonces tenemos:

$$\|S_J[\mathcal{P}_J](T_t f) - S_J[\mathcal{P}_J]f\|^2 \leq ((J+1) \|f\|^2 + \|f\|_w^2) C^2 2\alpha^{-1} 2^{-2J} |t|^2.$$

Tomando límite en ambos lados cuando J tiende a infinito, obtenemos:

$$\lim_{J \rightarrow \infty} \|S_J[\mathcal{P}_J](T_t f) - S_J[\mathcal{P}_J]f\|^2 \leq \lim_{J \rightarrow \infty} ((J+1) \|f\|^2 + \|f\|_w^2) C^2 2\alpha^{-1} 2^{-2J} |t|^2 = 0.$$

Por tanto, $\lim_{J \rightarrow \infty} \|S_J[\mathcal{P}_J](T_t f) - S_J[\mathcal{P}_J]f\| = 0$.

Finalmente, demostraremos que el límite anterior se verifica para todo $f \in L^2(\mathbb{R}^d)$, utilizando un argumento similar al del [Teorema 2.2](#). Dado que cualquier $f \in L^2(\mathbb{R}^d)$ puede aproximarse como el límite de una sucesión $\{f_n\}_{n \in \mathbb{N}}$, con $\|f_n\|_w < \infty$, y considerando que $S_J[\mathcal{P}_J]$ no

expande y T_t es unitario, usando la desigualdad triangular obtenemos que:

$$\|T_t(S_J[\mathcal{P}_J]f) - S_J[\mathcal{P}_J]f\| \leq \|T_t(S_J[\mathcal{P}_J]f_n) - S_J[\mathcal{P}_J]f_n\| + 2\|f - f_n\|.$$

De esta forma, al hacer que n tienda a infinito, se concluye que:

$$\lim_{J \rightarrow \infty} \|T_t(S_J[\mathcal{P}_J]f) - S_J[\mathcal{P}_J]f\| = 0.$$

□

2.5. Invarianza frente a pequeñas deformaciones

2.5.1. Cotas superiores en conmutadores de dispersión

Un difeomorfismo de \mathbb{R}^d lo suficientemente cercano a una traslación lleva x a $x - \tau(x)$, donde $\tau(x)$ es un campo de desplazamiento tal que $\|\text{Jac}(\tau)\|_\infty < 1$. La acción del difeomorfismo sobre $f \in L^2(\mathbb{R}^d)$ se define como $(T_\tau f)(x) = f(x - \tau(x))$. El máximo incremento de τ lo denotaremos por $\|\Delta\tau\|_\infty := \sup_{(x,u) \in \mathbb{R}^{2d}} |\tau(x) - \tau(u)|$.

Sea S_J un operador de dispersión de ventana calculado con una ondícula de dispersión que cumple la condición (2.8). Nuestro objetivo es establecer una cota superior para $\|S_J[\mathcal{P}_J](T_\tau f) - S_J[\mathcal{P}_J]f\|$ en términos de una norma de dispersión mixta $(\ell^1, L^2(\mathbb{R}^d))$:

$$\|U[\mathcal{P}_J]f\|_1 = \sum_{n=0}^{\infty} \|U[\Lambda_J^n]f\|.$$

De esta forma, veremos que el operador de dispersión de ventana es Lipschitz-continuo bajo la acción de difeomorfismos.

Denotaremos por $\mathcal{P}_{J,m}$ al subconjunto de \mathcal{P}_J de caminos de longitud estrictamente menor que m , y $(a \vee b) := \max(a, b)$. También denotaremos la norma suprema de un operador lineal A en $L^2(\mathbb{R}^d)$ por $\|A\|$ y al conmutador de dos operadores A y B por $[A, B] := AB - BA$.

Para alcanzar nuestro objetivo en esta sección, necesitamos dos lemas previos:

Lema 2.8. Para cualquier operador L en $L^2(\mathbb{R}^d)$ y cualquier $f \in L^2(\mathbb{R}^d)$, se cumple que:

$$\|[S_J[\mathcal{P}_J], L]f\| \leq \|U[\mathcal{P}_J]f\|_1 \|[U_J, L]\|.$$

Demostración. Si A y B son dos operadores, denotaremos $\{A, B\}$ como el operador definido por $\{A, B\}f = \{Af, Bf\}$. Introducimos un operador de módulo de ondícula sin promediar:

2. Teoría de Mallat sobre Redes de Dispersión

$$V_J f = \{|W[\lambda]f| = |f * \psi_\lambda|\}_{\lambda \in \Lambda_J}, \quad \text{con} \quad \Lambda_J = \{2^j r : j > -J, r \in G^+\},$$

y $U_J = \{A_J, V_J\}$. El propagador V_J genera todos los caminos $V_J(U[\Lambda_J^n]f) = U[\Lambda_J^{n+1}]f$ para cualquier $n \geq 0$. Dado que $U[\Lambda_J^0] = \text{Id}$, se deduce que $V_J^n = U[\Lambda_J^n]$. Para verificar la desigualdad que queremos, demostraremos que:

$$[S_J[\mathcal{P}_{J,m}], L] = \sum_{n=0}^m K_{m-n} V_J^n,$$

donde $K_n = \{[A_J, L], S_J[\mathcal{P}_{J,n-1}][V_J, L]\}$ satisface que:

$$\|K_n\| \leq \| [U_J, L] \|.$$

Dado que $V_J^n f = U[\Lambda_J^n]f$, esto implica que para cualquier $f \in L^2(\mathbb{R}^d)$:

$$\|[S_J[\mathcal{P}_{J,m}], L]f\| \leq \sum_{n=0}^m \|K_{m-n}\| \|V_J^n f\| \leq \| [U_J, L] \| \sum_{n=0}^{m-1} \|U[\Lambda_J^n]f\|,$$

y haciendo que m tienda a infinito en esta última expresión, demostraríamos la desigualdad buscada.

La propiedad $[S_J[\mathcal{P}_{J,m}], L] = \sum_{n=0}^m K_{m-n} V_J^n$ se demuestra comenzando por mostrar que:

$$S_J[\mathcal{P}_{J,m}]L = \{LA_J, S_J[\mathcal{P}_{J,m-1}]LV_J\} + K_m,$$

donde $K_m = \{[A_J, L], S_J[\mathcal{P}_{J,m-1}][V_J, L]\}$. En efecto, dado que $V_J^n = U[\Lambda_J^n]$, tenemos $A_J V_J^n = S_J[\Lambda_J^n]$ y $\mathcal{P}_{J,m} = \bigcup_{n=0}^{m-1} \Lambda_J^n$, lo que lleva a que $S_J[\mathcal{P}_{J,m}] = \{A_J V_J^n\}_{0 \leq n < m}$. Esto implica que:

$$\begin{aligned} S_J[\mathcal{P}_{J,m}]L &= \{A_J V_J^n L\}_{0 \leq n < m} \\ &= \{LA_J + [A_J, L], A_J V_J^{n-1} LV_J + A_J V_J^{n-1} [V_J, L]\}_{1 \leq n < m} \\ &= \{LA_J, S_J[\mathcal{P}_{J,m-1}]LV_J\} + \{[A_J, L], S_J[\mathcal{P}_{J,m-1}][V_J, L]\} \\ &= \{LA_J, S_J[\mathcal{P}_{J,m-1}]LV_J\} + K_m. \end{aligned}$$

Una sustitución de $S_J[\mathcal{P}_{J,m-1}]L$ en $S_J[\mathcal{P}_{J,m}]L = \{LA_J, S_J[\mathcal{P}_{J,m-1}]LV_J\} + K_m$ mediante la expresión derivada con esta misma fórmula nos da:

$$S_J[\mathcal{P}_{J,m}]L = \{LA_J, LA_JV_J, S_J[\mathcal{P}_{J,m-2}]LV_J^2\} + K_{m-1}V_J + K_m.$$

Con m sustituciones, obtenemos:

$$S_J[\mathcal{P}_{J,m}]L = \{LA_JV_J^n\}_{0 \leq n < m} + \sum_{n=0}^m K_{m-n}V_J^n = LS_J[\mathcal{P}_{J,m}] + \sum_{n=0}^m K_{m-n}V_J^n.$$

Vemos así que $[S_J[\mathcal{P}_{J,m}], L] = \sum_{n=0}^m K_{m-n}V_J^n$. Finalmente veamos que $\|K_m\| \leq \|U_J, L\|$, con $K_m = \{[A_J, L], S_J[\mathcal{P}_{J,m-1}][V_J, L]\}$. Dado que $S_J[\mathcal{P}_J]$ es no-expansivo, su restricción $S_J[\mathcal{P}_{J,m}]$ también lo es. Como $U_J = \{A_J, V_J\}$, tenemos que:

$$\begin{aligned} \|K_m f\|^2 &= \|[A_J, L]f\|^2 + \|S_J[\mathcal{P}_{J,m-1}][V_J, L]f\|^2 \\ &\leq \|[A_J, L]f\|^2 + \|[V_J, L]f\|^2 \\ &= \|[U_J, L]f\|^2 \\ &\leq \|[U_J, L]\|^2 \|f\|^2. \end{aligned}$$

□

Lema 2.9. Existe una constante $C > 0$ tal que, para todo $J \in \mathbb{Z}$, y todo $\tau \in C^2(\mathbb{R}^d)$, con $\|Jac(\tau)\|_\infty \leq \frac{1}{2}$, se cumple que:

$$\|[W_J, T_\tau]\| \leq C \left(\|Jac(\tau)\|_\infty \left(\log \left(\frac{\|\Delta\tau\|_\infty}{\|Jac(\tau)\|_\infty} \vee 1 \right) \right) + \|Hess(\tau)\|_\infty \right).$$

La demostración de este último lema se puede encontrar en ([Mal12], Apéndice E). No ha sido incluida en este trabajo debido a su elevada complejidad técnica.

2.5.2. Demostración de la invarianza frente a pequeñas deformaciones

Ya disponemos de todos los elementos necesarios para formular y probar el resultado clave de esta sección. Este resultado demuestra que el operador de dispersión de ventana que estamos construyendo, el cual recordamos que simula el comportamiento de una red neuronal convolucional, es Lipschitz-continuo bajo la acción de difeomorfismos, o dicho de otra forma, es invariante frente a pequeñas deformaciones.

Teorema 2.4. Existe una constante $C \in \mathbb{R}$ tal que, para todo $f \in L^2(\mathbb{R}^d)$, con $\|U[\mathcal{P}_J]f\|_1 < \infty$, y todo $\tau \in C^2(\mathbb{R}^d)$, con $\|Jac(\tau)\|_\infty \leq \frac{1}{2}$, se cumple que:

$$\|S_J[\mathcal{P}_J](T_\tau f) - S_J[\mathcal{P}_J]f\| \leq C\|U[\mathcal{P}_J]f\|_1 K(\tau),$$

2. Teoría de Mallat sobre Redes de Dispersión

con:

$$K(\tau) = 2^{-J} \|\tau\|_\infty + \|Jac(\tau)\|_\infty \left(\log \left(\frac{\|\Delta\tau\|_\infty}{\|Jac(\tau)\|_\infty} \vee 1 \right) \right) + \|Hess(\tau)\|_\infty,$$

y para todo $m \geq 0$:

$$\|S_J[\mathcal{P}_{J,m}](T_\tau f) - S_J[\mathcal{P}_{J,m}]f\| \leq Cm\|f\|K(\tau). \quad (2.13)$$

Demostración. Sea $[S_J[\mathcal{P}_J], T_\tau] = S_J[\mathcal{P}_J]T_\tau - T_\tau S_J[\mathcal{P}_J]$. Tenemos que:

$$\|S_J[\mathcal{P}_J](T_\tau f) - S_J[\mathcal{P}_J]f\| \leq \|T_\tau(S_J[\mathcal{P}_J]f) - S_J[\mathcal{P}_J]f\| + \| [S_J[\mathcal{P}_J], T_\tau]f\|.$$

El primer término a la derecha satisface:

$$\|T_\tau(S_J[\mathcal{P}_J]f) - S_J[\mathcal{P}_J]f\| \leq \|T_\tau A_J - A_J\| \|U[\mathcal{P}_J]f\|.$$

Dado que:

$$\|U[\mathcal{P}_J]f\| = \left(\sum_{m=0}^{\infty} \|U[\Lambda_J^m]f\|^2 \right)^{1/2} \leq \sum_{m=0}^{\infty} \|U[\Lambda_J^m]f\|,$$

tenemos que:

$$\|T_\tau(S_J[\mathcal{P}_J]f) - S_J[\mathcal{P}_J]f\| \leq \|T_\tau A_J - A_J\| \|U[\mathcal{P}_J]f\|_1.$$

Dado que $S_J[\mathcal{P}_J]$ itera sobre U_J , que es no-expansivo, el [Lema 2.8](#) demuestra la siguiente cota superior en los conmutadores de dispersión:

$$\|[S_J[\mathcal{P}_J], L]f\| \leq \|U[\mathcal{P}_J]f\|_1 \| [U_J, L] \|,$$

siendo L un operador en $L^2(\mathbb{R}^d)$. El operador $L = T_\tau$ también cumple con la siguiente propiedad:

$$\|[U_J, T_\tau]\| \leq \|[W_J, T_\tau]\|.$$

De hecho, $U_J = MW_J$, donde $M\{h_J, (h_\lambda)_{\lambda \in \Lambda_J}\} = \{h_J, (|h_\lambda|)_{\lambda \in \Lambda_J}\}$ es un operador módulo no-expansivo. Dado que $MT_\tau = T_\tau M$, tenemos que:

$$\|[U_J, T_\tau]\| = \|M_J[W_J, T_\tau]\| \leq \|[W_J, T_\tau]\|.$$

De esta forma, obtenemos:

$$\|S_J[\mathcal{P}_J](T_\tau f) - S_J[\mathcal{P}_J]f\| \leq \|U[\mathcal{P}_J]f\|_1 (\|T_\tau A_J - A_J\| + \|[W_J, T_\tau]\|).$$

El [Lema 2.7](#) muestra que $\|T_\tau A_J - A_J\| \leq C2^{-J}\|\tau\|_\infty$, que junto con esta última desigualdad, implican:

$$\|S_J[\mathcal{P}_J](T_\tau f) - S_J[\mathcal{P}_J]f\| \leq C\|U[\mathcal{P}_J]f\|_1 \left(2^{-J}\|\tau\|_\infty + \|[W_J, T_\tau]\|\right). \quad (2.14)$$

Para demostrar la desigualdad principal de este teorema, la dificultad radica en calcular una cota superior de $\|[W_J, T_\tau]\|$, y por tanto de $\|[W_J, T_\tau]\|^2 = \|[W_J, T_\tau]^*[W_J, T_\tau]\|$, donde A^* representa el adjunto de un operador A . Aplicando dicho comutador a f resulta:

$$[W_J, T_\tau]f = \{[A_J, T_\tau]f, ([W[\lambda], T_\tau]f)_{\lambda \in \Lambda_J}\},$$

y su norma es:

$$\|[W_J, T_\tau]f\|^2 = \|[A_J, T_\tau]f\|^2 + \sum_{\lambda \in \Lambda_J} \|[W[\lambda], T_\tau]f\|^2.$$

De esto se obtiene que:

$$[W_J, T_\tau]^*[W_J, T_\tau] = [A_J, T_\tau]^*[A_J, T_\tau] + \sum_{\lambda \in \Lambda_J} [W[\lambda], T_\tau]^*[W[\lambda], T_\tau].$$

El [Lema 2.9](#) demuestra que la norma del operador $[W_J, T_\tau]^*[W_J, T_\tau]$ está acotada superiormente ya que que:

$$\|[W_J, T_\tau]\| \leq C \left(\|Jac(\tau)\|_\infty \left(\log \left(\frac{\|\Delta\tau\|_\infty}{\|Jac(\tau)\|_\infty} \vee 1 \right) \right) + \|Hess(\tau)\|_\infty \right),$$

con $C > 0$ constante. Sustituyendo esta cota superior en (2.14) se demuestra la desigualdad principal del teorema. Se puede comprobar que dicha desigualdad se mantiene válida cuando se reemplaza \mathcal{P}_J por el subconjunto de caminos de longitud menor que m : $\mathcal{P}_{J,m} = \bigcup_{n < m} \Lambda_J^n$ si reemplazamos $\|U[\mathcal{P}_J]f\|_1$ por $\|U[\mathcal{P}_{J,m}]f\|_1$. La desigualdad (2.13) se deduce de:

2. Teoría de Mallat sobre Redes de Dispersión

$$\|U[\mathcal{P}_{J,m}]f\|_1 = \sum_{n=0}^{m-1} \|U[\Lambda_J^n]f\| \leq m\|f\|.$$

Esto se obtiene observando que:

$$\|U[\Lambda_J^n]f\| \leq \|U[\Lambda_J^{n-1}]f\| \leq \|f\|,$$

ya que $U[\Lambda_J^n]f$ se calcula aplicando el operador U_J , que preserva la norma, sobre $U[\Lambda_J^{n-1}]f$. \square

Este teorema demuestra que la distancia $\|S_J[\mathcal{P}_J](T_\tau f) - S_J[\mathcal{P}_J]f\|$ generada por la acción del difeomorfismo T_τ está acotada por un término de traslación proporcional a $2^{-J}\|\tau\|_\infty$ y un error de deformación proporcional a $\|Jac(\tau)\|_\infty$. Este error de deformación resulta del conmutador de la transformada de ondículas $[W_J, T_\tau]$. Concluimos así que nuestro operador de dispersión de ventana es invariante frente a pequeñas deformaciones.

3. Generalización de la Teoría de Mallat sobre Redes de Dispersión

Este capítulo está dedicado a extender y generalizar la teoría de redes de dispersión propuesta inicialmente por Stéphane Mallat. Mientras que el enfoque original de Mallat se centraba en el uso de ondículas y una arquitectura basada en convoluciones seguidas del operador módulo, aquí buscamos una formulación más general que permita mayor flexibilidad en la construcción de redes neuronales convolucionales para la extracción de características.

Comenzamos revisando los conceptos fundamentales de la teoría de Mallat. En particular, recordamos cómo las redes de dispersión logran invarianza por traslaciones utilizando transformadas de ondículas, y cómo la aplicación recursiva del módulo tras cada convolución juega un papel clave en esta arquitectura. Esto sirve de base para construir una teoría más general que capture las características esenciales de las redes convolucionales.

Posteriormente, formalizamos el concepto de un extractor de características convolucional general, que introduce varios tipos de no-linealidades y filtros más allá de las ondículas, abarcando funciones como las rectificadas lineales (ReLU) y sigmoides, así como diferentes tipos de operadores de pooling. Este extractor general es más flexible que el planteamiento original de Mallat y permite modelar arquitecturas más complejas que se utilizan habitualmente en redes neuronales profundas.

Finalmente, analizamos la invarianza vertical por traslaciones, una propiedad esencial en redes profundas, que permite que las características extraídas en capas superiores sean progresivamente más invariantes frente a traslaciones. Esto se demuestra formalmente en este capítulo, proporcionando una comprensión matemática sólida sobre cómo el pooling y la profundidad de la red influyen en dicha invarianza.

Este capítulo se basa en la investigación de Wiatowski y Bölcskei, siguiendo como referencia su trabajo, [WB18], junto con las contribuciones de otros autores que serán citados de forma adecuada.

3.1. Resumen de la teoría de Mallat sobre redes de dispersión

3.1.1. Definición del vector de características en redes de dispersión

Comenzamos haciendo un resumen de las redes de dispersión, introducidas en [Mal12] por Mallat y vistas en el [Capítulo 2](#). Dichas redes de dispersión son una arquitectura multinivel que involucra una transformada de ondículas seguida de la no-linealidad del módulo, sin aplicar posteriormente pooling. En particular, en [Definición 2.10](#) se definió el vector de características $\Phi_W(f)$ de la señal $f \in L^2(\mathbb{R}^d)$ como el conjunto:

$$\Phi_W(f) := \bigcup_{n=0}^{\infty} \Phi_W^n(f), \quad (3.1)$$

donde $\Phi_W^0(f) := \{f * \psi_{(-J,0)}\}$, y:

$$\Phi_W^n(f) := \left\{ (U[\underbrace{\lambda^{(j)}, \dots, \lambda^{(p)}}_{n \text{ índices}}]f) * \psi_{-J,0} \right\}_{\lambda^{(j)}, \dots, \lambda^{(p)} \in \Lambda_W \setminus \{-J,0\}},$$

para todo $n \in \mathbb{N}$, con:

$$U[\lambda^{(j)}, \dots, \lambda^{(p)}]f := \underbrace{\left| \cdots \left| f * \psi_{\lambda^{(j)}} \right| * \psi_{\lambda^{(k)}} \right| \cdots * \psi_{\lambda^{(p)}} \right|}_{n\text{-convoluciones seguidas de m\'odulo}}.$$

Aquí, el conjunto de índices Λ_W se define como:

$$\Lambda_W := \{(-J,0)\} \cup \{(j,k) \mid j \in \mathbb{Z}, j > -J, k \in \{0, \dots, K-1\}\},$$

e incluye los pares de escala j y direcciones k (de hecho, k es el índice de la dirección descrita por la matriz de rotación r_k), y:

$$\psi_\lambda(x) := 2^{dj} \psi(2^j r_k^{-1} x), \quad (3.2)$$

donde $\lambda = (j,k) \in \Lambda_W \setminus \{(-J,0)\}$ representa ondículas direccionales [Malo08], [AMVAo8], [Lee96], asociadas con la ondícula madre compleja $\psi \in L^1(\mathbb{R}^d) \cap L^2(\mathbb{R}^d)$. Los r_k , con $k \in \{0, \dots, K-1\}$, son elementos de un grupo finito de rotación G (si d es par, G es un subgrupo del grupo ortogonal especial $SO(d) := \{A \in \mathbb{R}^{d \times d} \mid A^T A = E \text{ y } \det(A) = 1\}$, donde $E \in \mathbb{R}^{d \times d}$ denota la matriz identidad; si d es impar, G es un subgrupo del grupo ortogonal $O(d) := \{A \in \mathbb{R}^{d \times d} \mid A^T A = E\}$; en este trabajo, nos hemos restringido anteriormente a $d = 1$ para una mejor comprensión). El índice $(-J,0) \in \Lambda_W$ está asociado con el filtro paso-bajo $\psi_{(-J,0)} \in L^1(\mathbb{R}^d) \cap L^2(\mathbb{R}^d)$, y $J \in \mathbb{Z}$ corresponde a la escala más gruesa resuelta por las ondículas direccionales (3.2).

3.1.2. Marcos semi-discretos

Definición 3.1. Sea $\{g_\lambda\}_{\lambda \in \Lambda} \subset L^1(\mathbb{R}^d) \cap L^2(\mathbb{R}^d)$ un conjunto de funciones indexado por un conjunto numerable Λ . La colección:

$$\Psi_\Lambda := \{T_b(Ig_\lambda)\}_{b \in \mathbb{R}^d, \lambda \in \Lambda},$$

es un *marco semi-discreto* para $L^2(\mathbb{R}^d)$ si existen constantes $A, B > 0$ tales que:

$$A\|f\|^2 \leq \sum_{\lambda \in \Lambda} \int_{\mathbb{R}^d} |\langle f, T_b(Ig_\lambda) \rangle|^2 db = \sum_{\lambda \in \Lambda} \|f * g_\lambda\|^2 \leq B\|f\|^2, \quad \forall f \in L^2(\mathbb{R}^d), \quad (3.3)$$

donde $\langle f, g \rangle := \int_{\mathbb{R}^d} f(x) \overline{g(x)} dx$, con $f, g \in L^2(\mathbb{R}^d)$, e $(If)(x) = \overline{f(-x)}$ denota la involución. Las funciones $\{g_\lambda\}_{\lambda \in \Lambda}$ son llamadas los *átomos* del marco Ψ_Λ y a los valores $\langle f, T_b(Ig_\lambda) \rangle = (f * g_\lambda)(b)$ se les denominan *coeficientes* del marco Ψ_Λ . Si $A = B$, el marco se denomina *marco ajustado*. Un marco ajustado con $A = 1$ se llama un *marco de Parseval*.

Se puede pensar en los marcos semi-discretos como marcos invariantes a traslaciones, con un parámetro de traslación continuo, y en el conjunto indexado numerable Λ como una colección de escalas, direcciones o desplazamientos de frecuencia, de ahí el término semi-discreto.

La familia de funciones $\{\psi_\lambda\}_{\lambda \in \Lambda_W}$ forma un marco de Parseval semi-discreto, definido como:

$$\Psi_{\Lambda_W} := \{T_b(I\psi_\lambda)\}_{b \in \mathbb{R}^d, \lambda \in \Lambda_W},$$

para $L^2(\mathbb{R}^d)$ [Malo8], [AAG93], [Kai94], y se satisface:

$$\sum_{\lambda \in \Lambda_W} \int_{\mathbb{R}^d} |\langle f, T_b(I\psi_\lambda) \rangle|^2 db = \sum_{\lambda \in \Lambda_W} \|f * \psi_\lambda\|^2 = \|f\|^2,$$

para cualquier $f \in L^2(\mathbb{R}^d)$, donde $\langle f, T_b(I\psi_\lambda) \rangle = (f * \psi_\lambda)(b)$ son los coeficientes del marco subyacente. Cabe destacar que, para un $\lambda \in \Lambda_W$, el parámetro de traslación $b \in \mathbb{R}^d$ puede tomar cualquier valor continuo dentro del espacio \mathbb{R}^d . Esto significa que, a diferencia de un marco discreto, donde b está limitado a un conjunto finito o numerable de valores, aquí existe un conjunto infinito continuo de posibles coeficientes de marco para cada λ . Este carácter continuo del parámetro b es una propiedad clave de los marcos semi-discretos y permite una mayor flexibilidad en la representación de señales. Véase Figura 3.1 para una ilustración en el dominio de la frecuencia de un marco semi-discreto de ondículas direccionales. Véase también ([WB18], Apéndice A) para una revisión breve de la teoría general de marcos semi-discretos, así como ([WB18], Apéndice B) y ([WB18], Apéndice C), donde se recogen ejemplos estructurados de marcos en 1-D y 2-D, respectivamente.

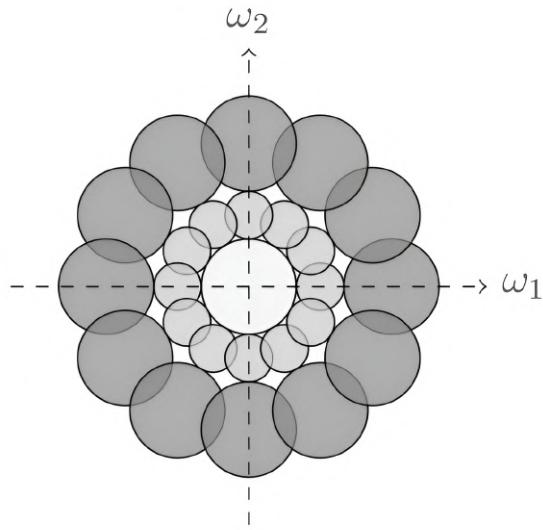


Figura 3.1.: Partición del plano de frecuencias \mathbb{R}^2 inducida por un marco semi-discreto de ondículas direccional, con $K = 12$ direcciones. Imagen extraída de [WB18].

3.1.3. Relación entre la arquitectura de redes de dispersión y las características extraídas

La arquitectura correspondiente al extractor de características Φ_W en la ecuación (3.1), ilustrada en Figura 3.2, es conocida como red de dispersión (*scattering network* en inglés) [Mal12], y emplea el marco Ψ_{Λ_W} y la no-linealidad del módulo $|\cdot|$ en cada capa de la red, pero no incluye pooling. Para $n \in \mathbb{N}$, el conjunto $\Phi_W^n(f)$ en (3.1) corresponde a las características de la función f generadas en la capa n -ésima de la red (véase Figura 3.2).

En el Teorema 2.3 demostramos que el extractor de características Φ_W es invariante por traslaciones en el sentido de:

$$\lim_{J \rightarrow \infty} \|\Phi_W(T_t f) - \Phi_W(f)\| = 0, \quad (3.4)$$

para todo $f \in L^2(\mathbb{R}^d)$ y $t \in \mathbb{R}^d$. Este resultado de invariancia es asintótico en el parámetro de escala $J \in \mathbb{Z}$ y no depende de la profundidad de la red, es decir, garantiza la invariancia total por traslaciones en cada capa de la red.

Además, en el Teorema 2.4 vimos que Φ_W también es estable con respecto a deformaciones de la forma $(T_\tau f)(x) := f(x - \tau(x))$, siendo $\tau \in C^2(\mathbb{R}^d)$ el campo de desplazamiento.

En la práctica, la clasificación de señales basada en redes de dispersión se lleva a cabo de la siguiente manera. Primero, la función f y los átomos de marco de ondículas $\{\psi_\lambda\}_{\lambda \in \Lambda_W}$ se

3.2. Construcción del extractor de características convolucional general

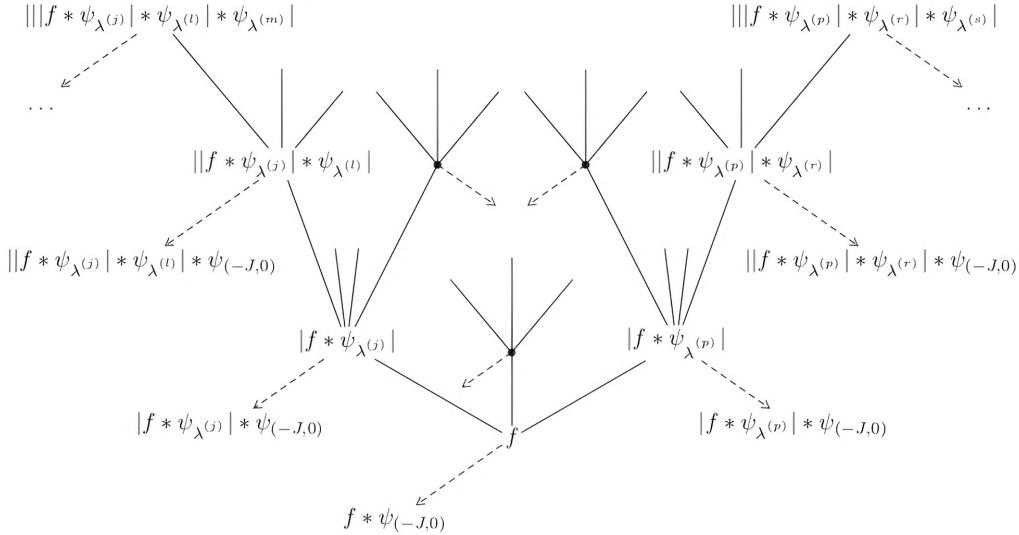


Figura 3.2.: Arquitectura de red de dispersión basada en filtros de ondículas y la no-linealidad del módulo. Los elementos del vector de características $\Phi_W(f)$ se indican en las puntas de las flechas. Imagen extraída de [WB18].

discretizan en vectores de dimensión finita. La red de dispersión resultante calcula entonces el vector de características $\Phi_W(f)$, cuya dimensión suele reducirse mediante un paso de mínimos cuadrados ortogonales [CCG91], y luego se introduce en un clasificador entrenable, como un SVM. Los resultados más avanzados de las redes de dispersión se han reportado en varias tareas de clasificación, como reconocimiento de dígitos manuscritos [BM13], discriminación de texturas [BM13], [Sif14], y clasificación de géneros musicales [AM14].

3.2. Construcción del extractor de características convolucional general

3.2.1. Diferencias y similitudes con las redes de dispersión

Como se mencionó previamente, las redes de dispersión siguen la arquitectura de las redes neuronales convolucionales [BCV13], [LBD⁺90], [LKF10], en el sentido de que utilizan convoluciones en cascada (con átomos $\{\psi_\lambda\}_{\lambda \in \Lambda_W}$ del marco de ondículas Ψ_{Λ_W} y no-linealidades como la función de módulo), pero sin aplicar un proceso de pooling. Las redes neuronales convolucionales generales estudiadas en la literatura exhiben características adicionales, tales como:

- Se emplea una amplia variedad de filtros, que incluyen tanto filtros aleatorios [HLo6], [RHBL07] como filtros aprendidos en un marco supervisado [JKRL09], o no supervisado [RHBL07], [RPCLO6].

3. Generalización de la Teoría de Mallat sobre Redes de Dispersión

- También se usan una variedad de funciones no-lineales, como las tangentes hiperbólicas [JKRL09], [RHBL07], unidades lineales rectificadas (ReLU) [GBB11], [NH10], y las sigmoides logísticas [MDH11], [GB10].
- Tras la aplicación de una convolución seguida de una no-linealidad, normalmente se emplea un operador de pooling, que puede ser submuestreo [PCD08], pooling promedio [HL06], [JKRL09], o max-pooling [SWP05], [ML06].
- Los filtros, las no-linealidades y los operadores de pooling pueden variar entre capas dentro de la misma red [LBH15], [GBC16].

Recordemos que el objetivo de este trabajo es desarrollar una teoría matemática de las redes neuronales convolucionales para la extracción de características, abarcando todos los aspectos mencionados anteriormente (exceptuando el max-pooling), y considerando que los operadores de pooling analizados son emulaciones en tiempo continuo de operadores de pooling en tiempo discreto.

Formalmente, en comparación con las redes de dispersión, en la capa n -ésima de la red, reemplazamos la operación módulo-ondícula $|f * \psi_\lambda|$ por una convolución con los átomos $g_{\lambda_n} \in L^1(\mathbb{R}^d) \cap L^2(\mathbb{R}^d)$ de un marco semi-discreto general $\Psi_{\Lambda_n} := \{T_b(Ig_{\lambda_n})\}_{b \in \mathbb{R}^d, \lambda_n \in \Lambda_n}$ para $L^2(\mathbb{R}^d)$, con un conjunto de índices numerable Λ_n , seguido de una no-linealidad $M_n : L^2(\mathbb{R}^d) \rightarrow L^2(\mathbb{R}^d)$ que cumple con la condición de Lipschitz $\|M_n f - M_n h\| \leq L_n \|f - h\|$, para todo $f, h \in L^2(\mathbb{R}^d)$, y $M_n f = 0$ para $f = 0$. El resultado de esta no-linealidad, $M_n(f * g_{\lambda_n})$, se somete luego a un proceso de pooling según:

$$f \mapsto S_n^{d/2} P_n(f)(S_n \cdot), \quad (3.5)$$

donde $S_n \geq 1$ es el factor de pooling y $P_n : L^2(\mathbb{R}^d) \rightarrow L^2(\mathbb{R}^d)$ satisface la condición de Lipschitz $\|P_n f - P_n h\| \leq R_n \|f - h\|$, para todo $f, h \in L^2(\mathbb{R}^d)$, y $P_n f = 0$ cuando $f = 0$. A continuación, comentamos los elementos individuales en nuestra arquitectura con más detalle. Los átomos del marco g_{λ_n} son arbitrarios y, por lo tanto, también pueden estructurarse como:

- Curvículas (*curvelets* en inglés): Funciones matemáticas diseñadas para representar eficientemente estructuras curvadas dentro de una señal o imagen, lo que las hace especialmente útiles en la detección de bordes suaves o curvaturas. Al igual que las ondículas, las curvículas se definen mediante una función madre que se traslada, rota y escala para cubrir toda la señal, pero con una mayor sensibilidad a las direcciones curvadas, lo que las convierte en una herramienta eficaz para analizar imágenes con detalles geométricos más complejos, como imágenes médicas o astronómicas.

- Crestículas (*ridgelets* en inglés): Funciones matemáticas que se utilizan para detectar estructuras lineales y crestas en imágenes. Están optimizadas para representar eficientemente señales o imágenes con discontinuidades lineales, como bordes o contornos alargados. Las crestículas son especialmente útiles en la compresión de imágenes y en aplicaciones donde las estructuras lineales juegan un papel clave, como en el análisis de imágenes de radar y tomografía.
- Ondículas (*wavelets* en inglés): Ya vistas en profundidad en [Capítulo 2](#), donde los átomos g_{λ_n} se obtienen a partir de una ondícula madre mediante operaciones de escalado y rotación.

3.2.2. Operadores de pooling

A continuación, explicamos cómo el operador de pooling en tiempo continuo (3.5) emula el pooling en tiempo discreto mediante submuestreo [PCDo8] o pooling promedio [HLo6], [JKRLog]. Consideremos una señal de tiempo discreto unidimensional $f_d \in \ell^2(\mathbb{Z}) := \{f_d : \mathbb{Z} \rightarrow \mathbb{C} \mid \sum_{k \in \mathbb{Z}} |f_d[k]|^2 < \infty\}$. El submuestreo por un factor $S \in \mathbb{N}$ en tiempo discreto se define por ([Vai93]), Sección 4):

$$f_d \mapsto h_d := f_d[S \cdot],$$

y equivale simplemente a retener cada muestra S -ésima de f_d . La transformada de Fourier en tiempo discreto de h_d se expresa como una suma de copias trasladadas y dilatadas de \widehat{f}_d , según ([Vai93]), Sección 4):

$$\widehat{h}_d(\theta) := \sum_{k \in \mathbb{Z}} h_d[k] e^{-2\pi i k \theta} = \frac{1}{S} \sum_{k=0}^{S-1} \widehat{f}_d\left(\frac{\theta - k}{S}\right). \quad (3.6)$$

Las copias trasladadas de \widehat{f}_d en (3.6) son una consecuencia de la periodicidad de la transformada discreta de Fourier. Por lo tanto, emulamos la operación de submuestreo en tiempo discreto en un tiempo continuo mediante la operación de dilatación:

$$f \mapsto h := S^{d/2} f(S \cdot), \quad f \in L^2(\mathbb{R}^d), \quad (3.7)$$

lo cual, en el dominio de frecuencia, equivale a la dilatación según $\widehat{h} = S^{-d/2} \widehat{f}(S^{-1} \cdot)$. El escalamiento por $S^{d/2}$ en (3.7) asegura la unitariedad de la operación de submuestreo en tiempo continuo. La operación general en (3.7) se ajusta a nuestra definición de pooling general, ya que puede recuperarse de (3.5) simplemente tomando P como la función identidad (que es, por supuesto, Lipschitz-continuo con constante $R = 1$ y satisface $\text{Id}f = 0$ para $f = 0$).

A continuación, consideraremos el pooling promedio. En tiempo discreto, el pooling promedio se define como:

3. Generalización de la Teoría de Mallat sobre Redes de Dispersión

$$f_d \mapsto h_d := (f_d * \phi_d)[S \cdot], \quad (3.8)$$

para el kernel de promedio $\phi_d \in \ell^2(\mathbb{Z})$, típicamente con soporte compacto, y el factor de promedio $S \in \mathbb{N}$. Tomando ϕ_d como una función ventana unitaria de longitud S , es decir:

$$\phi_d[k] = \begin{cases} 0, & \text{si } |k| \geq \frac{S}{2}, \\ 1, & \text{si } |k| < \frac{S}{2}, \end{cases}$$

el pooling promedio en tiempo descrito equivale a calcular promedios locales de S muestras consecutivas. Esto se realiza asignando un peso uniforme de $\frac{1}{S}$ a cada muestra dentro de la ventana de longitud S , de forma que todas las muestras contribuyen por igual al promedio. Los promedios ponderados, por otro lado, se obtienen modificando los valores del kernel $\phi_d[k]$ para asignar pesos específicos a cada muestra dentro de la ventana. Por ejemplo, un kernel gaussiano puede dar más peso a las muestras cercanas al centro de la ventana, mientras que las más alejadas tienen un peso menor. De esta forma, ϕ_d actúa como un filtro que determina la importancia relativa de cada muestra.

La operación (3.8) se puede emular en tiempo continuo según:

$$f \mapsto S^{d/2}(f * \phi)(S \cdot), \quad f \in L^2(\mathbb{R}^d), \quad (3.9)$$

con la ventana de promedio $\phi \in L^1(\mathbb{R}^d) \cap L^2(\mathbb{R}^d)$. Observemos que (3.9) se puede derivar de (3.5) tomando $P(f) = f * \phi$, $f \in L^2(\mathbb{R}^d)$. Gracias a la *desigualdad de Young*, la cual establece que para funciones $f \in L^p(\mathbb{R}^d)$ y $g \in L^r(\mathbb{R}^d)$, tal que $\|g\|_{L^r(\mathbb{R}^d)} = \|\tilde{g}\|_{L^r(\mathbb{R}^d)}$ y $\frac{1}{q} + 1 = \frac{1}{p} + \frac{1}{r}$, con $1 \leq p, q, r \leq \infty$ y $\tilde{g}(x) = g(x^{-1})$, $\forall x \in \mathbb{R}^d$, se cumple que $f * g$ existe y satisface que $\|f * g\|_{L^q(\mathbb{R}^d)} \leq \|f\|_{L^p(\mathbb{R}^d)} \|g\|_{L^r(\mathbb{R}^d)}$, podemos ver también que la convolución con ϕ es Lipschitz-continua con constante de Lipschitz $R = \|\phi\|_1$ y satisface trivialmente $Pf = 0$ para $f = 0$. En el resto de este trabajo, nos referiremos a la operación en (3.5) como *pooling de Lipschitz a través de dilatación* para indicar que (3.5) equivale esencialmente a la aplicación de una función Lipschitz-continua seguida de una dilatación en tiempo continuo. Sin embargo, cabe destacar que la operación dada en (3.5) no será unitaria en general.

A continuación, definimos los términos y recopilamos los resultados preliminares necesarios para el análisis del extractor de características convolucional general considerado. Los componentes básicos de esta red son las ternas (Ψ_n, M_n, P_n) asociadas con las capas individuales n de la red y denominadas *módulos*.

Definición 3.2. Para $n \in \mathbb{N}$, sea $\Psi_n = \{T_b(Ig_{\lambda_n})\}_{b \in \mathbb{R}^d, \lambda_n \in \Lambda_n}$ un marco semi-discreto para $L^2(\mathbb{R}^d)$ y sean $M_n : L^2(\mathbb{R}^d) \rightarrow L^2(\mathbb{R}^d)$ y $P_n : L^2(\mathbb{R}^d) \rightarrow L^2(\mathbb{R}^d)$ operadores Lipschitz-continuos con $M_n f = 0$ y $P_n f = 0$ para $f = 0$, respectivamente. Entonces, la secuencia de ternas:

$$\Omega := ((\Psi_n, M_n, P_n))_{n \in \mathbb{N}},$$

se denomina *secuencia de módulos*.

La siguiente definición introduce el concepto de caminos en conjuntos de índices, el cual nos será útil para formalizar la red de extracción de características. La idea de este formalismo es análoga al que hicimos en la [Subsección 2.2.3](#).

Definición 3.3. Sea $\Omega = ((\Psi_n, M_n, P_n))_{n \in \mathbb{N}}$ una secuencia de módulos, sean $\{g_{\lambda_n}\}_{\lambda_n \in \Lambda_n}$ los átomos del marco Ψ_n , y sea $S_n \geq 1$ el factor de pooling, de acuerdo a [\(3.5\)](#), asociado a la capa n -ésima de la red. Definimos el operador U_n asociado a la capa n -ésima de la red como $U_n : \Lambda_n \times L^2(\mathbb{R}^d) \rightarrow L^2(\mathbb{R}^d)$, tal que:

$$U_n(\lambda_n, f) := U_n[\lambda_n]f := S_n^{d/2} P_n(M_n(f * g_{\lambda_n})) (S_n \cdot). \quad (3.10)$$

Para $n \in \mathbb{N}$, definimos el conjunto $\Lambda_1^n := \Lambda_1 \times \Lambda_2 \times \cdots \times \Lambda_n$. Una secuencia ordenada $q = (\lambda_1, \lambda_2, \dots, \lambda_n) \in \Lambda_1^n$ se llama un *caminio*. Para el camino vacío establecemos $\Lambda_1^0 := \{\emptyset\}$ y $U_0[\emptyset]f := f$, para todo $f \in L^2(\mathbb{R}^d)$.

El operador U_n está bien definido, es decir, $U_n[\lambda_n]f \in L^2(\mathbb{R}^d)$ para todo $(\lambda_n, f) \in \Lambda_n \times L^2(\mathbb{R}^d)$, gracias a que:

$$\begin{aligned} \|U_n[\lambda_n]f\|^2 &= S_n^d \int_{\mathbb{R}^d} |P_n(M_n(f * g_{\lambda_n})) (S_n x)|^2 dx \\ &= \int_{\mathbb{R}^d} |P_n(M_n(f * g_{\lambda_n})) (y)|^2 dy \\ &= \|P_n(M_n(f * g_{\lambda_n}))\|^2 \leq R_n^2 \|M_n(f * g_{\lambda_n})\|^2 \end{aligned} \quad (3.11)$$

$$\leq L_n^2 R_n^2 \|f * g_{\lambda_n}\|^2 \leq B_n L_n^2 R_n^2 \|f\|^2. \quad (3.12)$$

Para la desigualdad en [\(3.11\)](#), utilizamos la Lipschitz-continuidad de P_n de acuerdo a $\|P_n f - P_n h\|^2 \leq R_n^2 \|f - h\|^2$, con $P_n h = 0$ para $h = 0$, obteniendo $\|P_n f\|^2 \leq R_n^2 \|f\|^2$. Argumentos similares nos llevan a la primera desigualdad en [\(3.12\)](#). El último paso en [\(3.12\)](#) se debe a:

$$\|f * g_{\lambda_n}\|^2 \leq \sum_{\lambda'_n \in \Lambda_n} \|f * g_{\lambda'_n}\|^2 \leq B_n \|f\|^2,$$

que resulta de la condición de marco [\(3.3\)](#) en Ψ_n . También necesitaremos la extensión del operador U_n a caminos $q \in \Lambda_1^n$ según la siguiente fórmula:

3. Generalización de la Teoría de Mallat sobre Redes de Dispersión

$$\begin{aligned} U[q]f &= U[(\lambda_1, \lambda_2, \dots, \lambda_n)]f \\ &:= U_n[\lambda_n] \cdots U_2[\lambda_2]U_1[\lambda_1]f, \end{aligned} \quad (3.13)$$

con $U[\emptyset]f := f$. Notemos que la operación multietapa (3.13) está nuevamente bien definida ya que:

$$\|U[q]f\|^2 \leq \left(\prod_{k=1}^n B_k L_k^2 R_k^2 \right) \|f\|^2, \quad (3.14)$$

para $q \in \Lambda_1^n$ y $f \in L^2(\mathbb{R}^d)$, lo cual se deduce de la aplicación repetida de (3.12).

En las redes de dispersión, un átomo ψ_λ , $\lambda \in \Lambda_W$, del marco de ondículas Ψ_{Λ_W} , es decir, el filtro paso-bajo $\psi_{(-J,0)}$, se utiliza para generar las características extraídas según (3.1) (véase Figura 3.2). Siguiendo esta construcción, designamos uno de los átomos en cada marco de la secuencia de módulos $\Omega = ((\Psi_n, M_n, P_n))_{n \in \mathbb{N}}$ como el *átomo generador de salida* $\chi_{n-1} := g_{\lambda_n^*}, \lambda_n^* \in \Lambda_n$, de la capa $(n-1)$ -ésima. Los átomos $\{g_{\lambda_n}\}_{\lambda_n \in \Lambda_n \setminus \{\lambda_n^*\}} \cup \{\chi_{n-1}\}$ en Ψ_n se usan entonces a través de dos capas consecutivas en el sentido de que $\chi_{n-1} = g_{\lambda_n^*}$ genera la salida en la capa $(n-1)$ -ésima, y los $\{g_{\lambda_n}\}_{\lambda_n \in \Lambda_n \setminus \{\lambda_n^*\}}$ propagan las señales de la capa $(n-1)$ -ésima a la capa n -ésima según (3.10) (véase Figura 3.3).

Obsérvese que esta teoría no requiere que los átomos generadores de salida sean filtros de paso bajo. A partir de ahora, con un pequeño abuso de notación, escribiremos Λ_n para $\Lambda_n \setminus \{\lambda_n^*\}$. Finalmente, cabe destacar que extraer características en cada capa de red usando un átomo generador de salida puede considerarse como el uso de conexiones de salto de capa [HZRS15], que omiten capas de red más abajo y luego alimentan las señales propagadas al vector de características.

Ya estamos listos para definir el extractor de características Φ_Ω basado en la secuencia de módulos Ω .

Definición 3.4. Sea $\Omega = ((\Psi_n, M_n, P_n))_{n \in \mathbb{N}}$ una secuencia de módulos. El *extractor de características* Φ_Ω basado en Ω lleva $L^2(\mathbb{R}^d)$ a su vector de características:

$$\Phi_\Omega(f) := \bigcup_{n=0}^{\infty} \Phi_\Omega^n(f), \quad (3.15)$$

donde $\Phi_\Omega^n(f) := \{(U[q]f) * \chi_n\}_{q \in \Lambda_1^n}, \forall n \in \mathbb{N}$.

El conjunto $\Phi_\Omega^n(f)$ en la ecuación (3.15) corresponde a las características de la función f generadas en la capa n -ésima de la red (véase Figura 3.3), donde $n = 0$ corresponde a la raíz de la red. El extractor de características $\Phi_\Omega : L^2(\mathbb{R}^d) \rightarrow (L^2(\mathbb{R}^d))^Q$, con $Q := \bigcup_{n=0}^{\infty} \Lambda_1^n$,

3.2. Construcción del extractor de características convolucional general

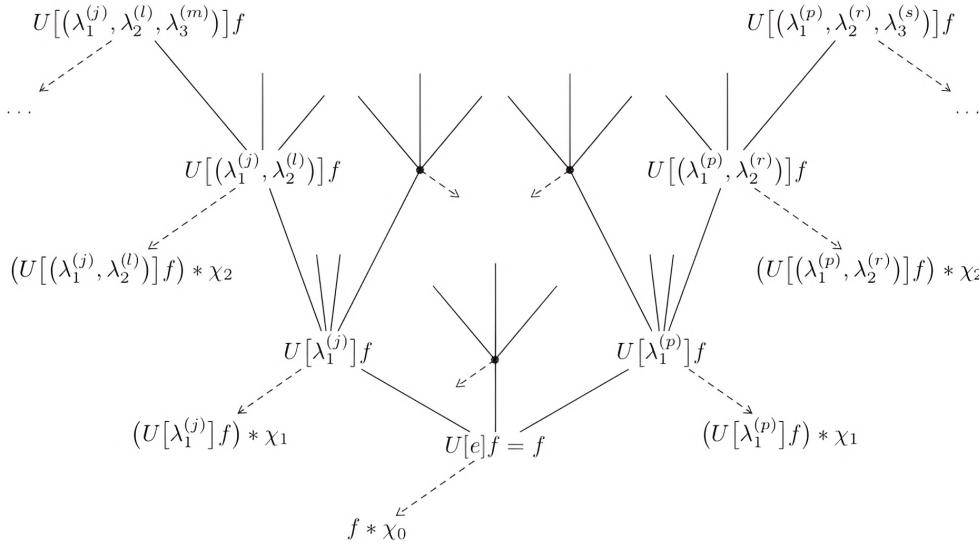


Figura 3.3.: Arquitectura de red subyacente al extractor de características convolucional general. El índice $\lambda_n^{(k)}$ corresponde al k -ésimo átomo $g_{\lambda_n^{(k)}}$ del marco Ψ_n asociado a la capa n -ésima de la red. La función χ_n es el átomo generador de salida de la capa n -ésima. Imagen extraída de [WB18].

está bien definido, es decir, $\Phi_\Omega(f) \in (L^2(\mathbb{R}^d))^\mathcal{Q}$, para todo $f \in L^2(\mathbb{R}^d)$, bajo una condición técnica sobre la secuencia de módulos Ω , formalizada a continuación.

Proposición 3.1. *Sea $\Omega = ((\Psi_n, M_n, P_n))_{n \in \mathbb{N}}$ una secuencia de módulos. Denotemos las cotas superiores del marco Ψ_n por $B_n > 0$ y las constantes de Lipschitz de los operadores M_n y P_n por $L_n > 0$ y $R_n > 0$, respectivamente. Si:*

$$\max\{B_n, B_n L_n^2 R_n^2\} \leq 1, \quad \forall n \in \mathbb{N}, \quad (3.16)$$

entonces el extractor de características $\Phi_\Omega : L^2(\mathbb{R}^d) \rightarrow (L^2(\mathbb{R}^d))^\mathcal{Q}$ está bien definido, es decir, $\Phi_\Omega(f) \in (L^2(\mathbb{R}^d))^\mathcal{Q}$ para todo $f \in L^2(\mathbb{R}^d)$.

Demostración. Tenemos que demostrar que $\Phi_\Omega(f) \in (L^2(\mathbb{R}^d))^\mathcal{Q}$, para todo $f \in L^2(\mathbb{R}^d)$. Esto se logrará demostrando un resultado aún más fuerte:

$$\|\Phi_\Omega(f)\| \leq \|f\|, \quad \forall f \in L^2(\mathbb{R}^d), \quad (3.17)$$

lo que, dado que $\|f\| < \infty$, demuestra nuestra afirmación inicial. Para simplificar la notación, definimos $f_q := U[q]f$, para $f \in L^2(\mathbb{R}^d)$. Debido a (3.14) y (3.16), tenemos que $\|f_q\| \leq \|f\| < \infty$, y por lo tanto $f_q \in L^2(\mathbb{R}^d)$. La idea clave de la prueba es similar a la demostración de la Proposición 2.3, y se basa en emplear de manera meticulosa un argumento

3. Generalización de la Teoría de Mallat sobre Redes de Dispersión

de series telescopicas. Comenzamos escribiendo:

$$\begin{aligned}\|\Phi_\Omega(f)\|^2 &= \sum_{n=0}^{\infty} \sum_{q \in \Lambda_1^n} \|f_q * \chi_n\|^2 \\ &= \lim_{N \rightarrow \infty} \sum_{n=0}^N \underbrace{\sum_{q \in \Lambda_1^n} \|f_q * \chi_n\|^2}_{:= a_n},\end{aligned}\tag{3.18}$$

El paso clave es demostrar que a_n puede ser acotado superiormente de la siguiente forma:

$$a_n \leq b_n - b_{n+1}, \quad \forall n \in \mathbb{N}_0,\tag{3.19}$$

con $b_n := \sum_{q \in \Lambda_1^n} \|f_q\|^2$, y usar este resultado en un argumento de series telescopicas como sigue:

$$\begin{aligned}\sum_{n=0}^N a_n &\leq \sum_{n=0}^N (b_n - b_{n+1}) \\ &= (b_0 - b_1) + (b_1 - b_2) + \cdots + (b_N - b_{N+1}) \\ &= b_0 - \underbrace{b_{N+1}}_{\geq 0} \\ &\leq b_0 = \sum_{q \in \Lambda_1^0} \|f_q\|^2 = \|U[\emptyset]f\|^2 = \|f\|^2.\end{aligned}$$

Tenemos que (3.18) implica (3.17). Comenzamos observando que (3.19) se lee como:

$$\sum_{q \in \Lambda_1^n} \|f_q * \chi_n\|^2 \leq \sum_{q \in \Lambda_1^n} \|f_q\|^2 - \sum_{q \in \Lambda_1^{n+1}} \|f_q\|^2,\tag{3.20}$$

para todo $n \in \mathbb{N}_0$. Procedemos a examinar el segundo término en el lado derecho de (3.20). Cada camino

$$\tilde{q} \in \Lambda_1^{n+1} = \underbrace{\Lambda_1 \times \cdots \times \Lambda_n}_{=\Lambda_1^n} \times \Lambda_{n+1},$$

de longitud $n+1$, puede descomponerse en un camino $q \in \Lambda_1^n$ de longitud n y un índice $\lambda_{n+1} \in \Lambda_{n+1}$, según $\tilde{q} = (q, \lambda_{n+1})$. Por (3.13), tenemos que:

$$U[\tilde{q}] = U[(q, \lambda_{n+1})] = U_{n+1}[\lambda_{n+1}]U[q],$$

lo que implica:

$$\sum_{\tilde{q} \in \Lambda_1^{n+1}} \|f_{\tilde{q}}\|^2 = \sum_{q \in \Lambda_1^n} \sum_{\lambda_{n+1} \in \Lambda_{n+1}} \|U_{n+1}[\lambda_{n+1}]f_q\|^2. \quad (3.21)$$

Sustituyendo el segundo término en el lado derecho de (3.20) por (3.21), obtenemos:

$$\sum_{q \in \Lambda_1^n} \|f_q * \chi_n\|^2 \leq \sum_{q \in \Lambda_1^n} \left(\|f_q\|^2 - \sum_{\lambda_{n+1} \in \Lambda_{n+1}} \|U_{n+1}[\lambda_{n+1}]f_q\|^2 \right), \quad \forall n \in \mathbb{N}_0,$$

lo que puede reescribirse como:

$$\sum_{q \in \Lambda_1^n} \left(\|f_q * \chi_n\|^2 + \sum_{\lambda_{n+1} \in \Lambda_{n+1}} \|U_{n+1}[\lambda_{n+1}]f_q\|^2 \right) \leq \sum_{q \in \Lambda_1^n} \|f_q\|^2, \quad \forall n \in \mathbb{N}_0. \quad (3.22)$$

A continuación, observemos que el segundo término dentro de la suma del lado izquierdo de (3.22) se puede escribir como:

$$\begin{aligned} \sum_{\lambda_{n+1} \in \Lambda_{n+1}} \|U_{n+1}[\lambda_{n+1}]f_q\|^2 &= \sum_{\lambda_{n+1} \in \Lambda_{n+1}} \int_{\mathbb{R}^d} |(U_{n+1}[\lambda_{n+1}]f_q)(x)|^2 dx \\ &= \sum_{\lambda_{n+1} \in \Lambda_{n+1}} S_{n+1}^d \int_{\mathbb{R}^d} |P_{n+1}(M_{n+1}(f_q * g_{\lambda_{n+1}}))(S_{n+1}x)|^2 dx \\ &= \sum_{\lambda_{n+1} \in \Lambda_{n+1}} \int_{\mathbb{R}^d} |P_{n+1}(M_{n+1}(f_q * g_{\lambda_{n+1}}))(y)|^2 dy \\ &= \sum_{\lambda_{n+1} \in \Lambda_{n+1}} \|P_{n+1}(M_{n+1}(f_q * g_{\lambda_{n+1}}))\|^2, \end{aligned} \quad (3.23)$$

para todo $n \in \mathbb{N}_0$. Nótese que $f_q \in L^2(\mathbb{R}^d)$, como se ha establecido antes, y $g_{\lambda_{n+1}} \in L^1(\mathbb{R}^d)$ por suposición, cumplen que $(f_q * g_{\lambda_{n+1}}) \in L^2(\mathbb{R}^d)$ gracias a la desigualdad de Young. Usamos la condición de Lipschitz de M_{n+1} y P_{n+1} , es decir, $\|M_{n+1}(f_q * g_{\lambda_{n+1}}) - M_{n+1}h\| \leq L_{n+1}\|f_q * g_{\lambda_{n+1}} - h\|$, y $\|P_{n+1}(f_q * g_{\lambda_{n+1}}) - P_{n+1}h\| \leq R_{n+1}\|f_q * g_{\lambda_{n+1}} - h\|$, junto con $M_{n+1}h = 0$ y $P_{n+1}h = 0$ para $h = 0$, para acotar por encima el término de dentro de la suma en (3.23) de acuerdo con:

3. Generalización de la Teoría de Mallat sobre Redes de Dispersión

$$\begin{aligned}\|P_{n+1}(M_{n+1}(f_q * g_{\lambda_{n+1}}))\| &\leq R_{n+1}^2 \|M_{n+1}(f_q * g_{\lambda_{n+1}})\|^2 \\ &\leq L_{n+1}^2 R_{n+1}^2 \|f_q * g_{\lambda_{n+1}}\|^2, \quad \forall n \in \mathbb{N}_0.\end{aligned}\quad (3.24)$$

Sustituyendo el segundo término dentro de la suma en el lado izquierdo de (3.22) por la cota superior resultante de la inserción de (3.24) en (3.23), obtenemos:

$$\begin{aligned}&\sum_{q \in \Lambda_1^n} \left(\|f_q * \chi_n\|^2 + L_{n+1}^2 R_{n+1}^2 \sum_{\lambda_{n+1} \in \Lambda_{n+1}} \|f_q * g_{\lambda_{n+1}}\|^2 \right) \\ &\leq \sum_{q \in \Lambda_1^n} \max \left\{ 1, L_{n+1}^2 R_{n+1}^2 \right\} \left(\|f_q * \chi_n\|^2 + \sum_{\lambda_{n+1} \in \Lambda_{n+1}} \|f_q * g_{\lambda_{n+1}}\|^2 \right), \quad \forall n \in \mathbb{N}_0.\end{aligned}\quad (3.25)$$

Dado que las funciones $g_{\lambda_{n+1}}$ y χ_n son átomos del marco semi-discreto Ψ_{n+1} para $L^2(\mathbb{R}^d)$ y que $f_q \in L^2(\mathbb{R}^d)$, como se estableció previamente, tenemos que:

$$\|f_q * \chi_n\|^2 + \sum_{\lambda_{n+1} \in \Lambda_{n+1}} \|f_q * g_{\lambda_{n+1}}\|^2 \leq B_{n+1} \|f_q\|^2,$$

lo que al usarse en (3.25) resulta:

$$\begin{aligned}&\sum_{q \in \Lambda_1^n} \left(\|f_q * \chi_n\|^2 + \sum_{\lambda_{n+1} \in \Lambda_{n+1}} \|U_{n+1}[\lambda_{n+1}]f_q\|^2 \right) \\ &\leq \sum_{q \in \Lambda_1^n} \max \left\{ 1, L_{n+1}^2 R_{n+1}^2 \right\} B_{n+1} \|f_q\|^2 \\ &= \sum_{q \in \Lambda_1^n} \max \left\{ B_{n+1}, B_{n+1} L_{n+1}^2 R_{n+1}^2 \right\} \|f_q\|^2,\end{aligned}\quad (3.26)$$

para todo $n \in \mathbb{N}_0$. Finalmente, invocando la suposición:

$$\max \{B_n, B_n L_n^2 R_n^2\} \leq 1, \quad \forall n \in \mathbb{N},$$

en (3.26), obtenemos (3.22), lo cual completa la demostración. \square

Dado que la condición (3.16) es de gran importancia, la formalizamos de la siguiente manera.

Definición 3.5. Sea $\Omega = ((\Psi_n, M_n, P_n))_{n \in \mathbb{N}}$ una secuencia de módulos, con cotas superiores del marco $B_n > 0$ y constantes de Lipschitz $L_n, R_n > 0$ para los operadores M_n y P_n , respectivamente. La condición:

$$\max\{B_n, B_n L_n^2 R_n^2\} \leq 1, \quad \forall n \in \mathbb{N}, \quad (3.27)$$

se denomina *condición de admisibilidad para secuencias de módulos*. Las secuencias de módulos que satisfacen (3.27) se denominan *admisibles*.

La condición (3.27) se cumple fácilmente en la práctica. Para demostrar esto, primero observamos que B_n está determinado por el marco Ψ_n (por ejemplo, el marco de ondículas direccionales visto en la Sección 3.1 tiene $B = 1$). L_n está definido por la no-linealidad M_n (por ejemplo, la función módulo $M = |\cdot|$ tiene $L = 1$), y R_n depende del operador P_n dado en (3.5) (por ejemplo, hacer submuestreo equivale a $P = Id$ y tiene $R = 1$). Obviamente, la condición (3.27) se cumple si:

$$B_n \leq \min\left\{1, L_n^{-2} R_n^{-2}\right\}, \quad \forall n \in \mathbb{N},$$

lo cual se puede conseguir normalizando adecuadamente los elementos del marco Ψ_n , sin afectar ni a la invarianza vertical por traslaciones ni a la invarianza frente a pequeñas deformaciones.

3.3. Invarianza vertical por traslaciones

El siguiente teorema establece que, bajo condiciones de decaimiento muy leves en las transformadas de Fourier $\widehat{\chi}_n$ de los átomos generadores de salidas χ_n , el extractor de características Φ_Ω presenta invarianza vertical por traslaciones, en el sentido de que las características se vuelven más invariantes a medida que la profundidad de la red aumenta. Este resultado concuerda con observaciones empíricas en la literatura de Aprendizaje Profundo, por ejemplo, en [HLo6], [SWPo5] y [MLo6], donde se argumenta que las salidas generadas en capas más profundas tienden a ser más invariantes a traslaciones.

Teorema 3.1. Sea $\Omega = ((\Psi_n, M_n, P_n))_{n \in \mathbb{N}}$ una secuencia de módulos admisible, sea $S_n \geq 1$, $n \in \mathbb{N}$, el factor de pooling en (3.10), y supongamos que los operadores $M_n : L^2(\mathbb{R}^d) \rightarrow L^2(\mathbb{R}^d)$ y $P_n : L^2(\mathbb{R}^d) \rightarrow L^2(\mathbb{R}^d)$ conmutan con el operador de traslación T_t , es decir:

$$M_n(T_t f) = T_t(M_n f), \quad P_n(T_t f) = T_t(P_n f), \quad (3.28)$$

para todo $f \in L^2(\mathbb{R}^d)$, $t \in \mathbb{R}^d$, y $n \in \mathbb{N}$. Entonces:

- (1) Las características $\Phi_\Omega^n(f)$ generadas en la capa n -ésima de la red satisfacen:

$$\Phi_\Omega^n(T_t f) = T_{t/(S_1 \dots S_n)}(\Phi_\Omega^n(f)), \quad (3.29)$$

3. Generalización de la Teoría de Mallat sobre Redes de Dispersión

para todo $f \in L^2(\mathbb{R}^d)$, $t \in \mathbb{R}^d$, y $n \in \mathbb{N}$, donde $T_t(\Phi_\Omega^n(f))$ se refiere a la aplicación por elementos de T_t , es decir, $T_t(\Phi_\Omega^n(f)) := \{T_t h \mid \forall h \in \Phi_\Omega^n(f)\}$.

- (ii) Si, además, existe una constante $K > 0$ (que no depende de n) tal que las transformadas de Fourier $\widehat{\chi}_n$ de los átomos generadores de salidas χ_n satisfacen la condición de decadimiento:

$$|\widehat{\chi}_n(\omega)| |\omega| \leq K, \quad c.p.d. \omega \in \mathbb{R}^d, \quad \forall n \in \mathbb{N}_0, \quad (3.30)$$

entonces:

$$\|\Phi_\Omega^n(T_t f) - \Phi_\Omega^n(f)\| \leq \frac{2\pi|t|K}{S_1 \cdots S_n} \|f\|, \quad (3.31)$$

para todo $f \in L^2(\mathbb{R}^d)$ y $t \in \mathbb{R}^d$.

Demostración. Comenzamos probando (I). El paso clave para establecer (3.29) es demostrar que el operador U_n , $n \in \mathbb{N}$, definido en (3.10), satisface la relación:

$$U_n[\lambda_n](T_t f) = T_{t/S_n}(U_n[\lambda_n]f), \quad (3.32)$$

para todo $f \in L^2(\mathbb{R}^d)$, $t \in \mathbb{R}^d$ y $\lambda_n \in \Lambda_n$. Con la definición de $U[q]$ en (3.13), esto se convierte en:

$$U[q](T_t f) = T_{t/(S_1 \cdots S_n)}(U[q]f), \quad (3.33)$$

para todo $f \in L^2(\mathbb{R}^d)$, $t \in \mathbb{R}^d$ y $q \in \Lambda_1^n$. La identidad (3.29) es una consecuencia directa de (3.33) y la covarianza de traslación del operador de convolución:

$$\begin{aligned} \Phi_\Omega^n(T_t f) &= \{(U[q](T_t f)) * \chi_n\}_{q \in \Lambda_1^n} \\ &= \left\{ \left(T_{t/(S_1 \cdots S_n)}(U[q]f) \right) * \chi_n \right\}_{q \in \Lambda_1^n} \\ &= T_{t/(S_1 \cdots S_n)} \{(U[q]f) * \chi_n\}_{q \in \Lambda_1^n} \\ &= T_{t/(S_1 \cdots S_n)} \Phi_\Omega^n(f), \quad \forall f \in L^2(\mathbb{R}^d), \forall t \in \mathbb{R}^d. \end{aligned}$$

Para establecer (3.32), primero definimos el operador unitario:

$$D_n : L^2(\mathbb{R}^d) \rightarrow L^2(\mathbb{R}^d), \quad D_n f := S_n^{d/2} f(S_n \cdot),$$

y observamos que:

$$\begin{aligned}
 U_n[\lambda_n](T_t f) &= S_n^{d/2} P_n(M_n((T_t f) * g_{\lambda_n})) (S_n \cdot) \\
 &= D_n(P_n(M_n((T_t f) * g_{\lambda_n}))) \\
 &= D_n(P_n(M_n(T_t(f * g_{\lambda_n})))) \\
 &= D_n(P_n(T_t(M_n(f * g_{\lambda_n})))) \tag{3.34}
 \end{aligned}$$

$$= D_n(T_t(P_n(M_n(f * g_{\lambda_n})))), \tag{3.35}$$

para todo $f \in L^2(\mathbb{R}^d)$ y $t \in \mathbb{R}^d$, donde en (3.34) y (3.35) usamos:

$$M_n T_t = T_t M_n, \quad \text{y} \quad P_n T_t = T_t P_n,$$

para todo $n \in \mathbb{N}$ y $t \in \mathbb{R}^d$, respectivamente. Luego, usando:

$$\begin{aligned}
 D_n(T_t f) &= S_n^{d/2} f(S_n \cdot -t) \\
 &= S_n^{d/2} f(S_n(\cdot - t/S_n)) \\
 &= T_{t/S_n}(D_n f), \quad \forall f \in L^2(\mathbb{R}^d), \quad \forall t \in \mathbb{R}^d,
 \end{aligned}$$

en (3.35) resulta que:

$$\begin{aligned}
 U_n[\lambda_n](T_t f) &= D_n(T_t(P_n(M_n(f * g_{\lambda_n})))) \\
 &= T_{t/S_n}(D_n(P_n(M_n(f * g_{\lambda_n})))) \\
 &= T_{t/S_n}(U_n[\lambda_n]f),
 \end{aligned}$$

para todo $f \in L^2(\mathbb{R}^d)$ y $t \in \mathbb{R}^d$. Esto completa la prueba de (I).

A continuación, probamos (II). Para facilitar la notación, nuevamente, definimos $f_q := U[q]f$, para $f \in L^2(\mathbb{R}^d)$. Gracias a (3.14) y la condición de admisibilidad para secuencias de módulos (3.27) tenemos que $\|f_q\| \leq \|f\| < \infty$, y por lo tanto, $f_q \in L^2(\mathbb{R}^d)$. Primero escribimos:

$$\|\Phi_\Omega^n(T_t f) - \Phi_\Omega^n(f)\|^2 = \|T_{t/(S_1 \dots S_n)}(\Phi_\Omega^n(f)) - \Phi_\Omega^n(f)\|^2 \quad (3.36)$$

$$\begin{aligned} &= \sum_{q \in \Lambda_1^n} \|T_{t/(S_1 \dots S_n)}(f_q * \chi_n) - (f_q * \chi_n)\|^2 \\ &= \sum_{q \in \Lambda_1^n} \|M_{-t/(S_1 \dots S_n)}(\widehat{f_q * \chi_n}) - (\widehat{f_q * \chi_n})\|^2, \end{aligned} \quad (3.37)$$

para todo $n \in \mathbb{N}$, donde en (3.36) usamos (3.29), y en (3.37) aplicamos la la relación $\widehat{T_t f} = M_{-t} \widehat{f}$, $\forall f \in L^2(\mathbb{R}^d)$, $\forall t \in \mathbb{R}^d$, junto con la *fórmula de Parseval* (nótese que $(f_q * \chi_n) \in L^2(\mathbb{R}^d)$ gracias a la desigualdad de Young), la cual establece que, para funciones $f \in L^2(\mathbb{R}^d)$ y $g \in L^2(\mathbb{R}^d)$, se tiene $\int_{\mathbb{R}^d} f(x) \overline{g(x)} dx = \int_{\mathbb{R}^d} \widehat{f}(\omega) \overline{\widehat{g}(\omega)} d\omega$, y en el caso particular $g = f$, se tiene que $\|f\| = \|\widehat{f}\|$.

El paso clave ahora es establecer la cota superior:

$$\|M_{-t/(S_1 \dots S_n)}(\widehat{f_q * \chi_n}) - (\widehat{f_q * \chi_n})\|^2 \leq \frac{4\pi^2 |t|^2 K^2}{(S_1 \dots S_n)^2} \|f_q\|^2, \quad \forall n \in \mathbb{N}, \quad (3.38)$$

donde $K > 0$ corresponde a la constante en la condición de decaimiento (3.30). Tenemos que:

$$\sum_{q \in \Lambda_1^n} \|f_q\|^2 \leq \sum_{q \in \Lambda_1^{n-1}} \|f_q\|^2, \quad \forall n \in \mathbb{N}, \quad (3.39)$$

lo cual sigue de (3.19) gracias a:

$$\begin{aligned} 0 &\leq \sum_{q \in \Lambda_1^{n-1}} \|f_q * \chi_{n-1}\|^2 = a_{n-1} \leq b_{n-1} - b_n \\ &= \sum_{q \in \Lambda_1^{n-1}} \|f_q\|^2 - \sum_{q \in \Lambda_1^n} \|f_q\|^2, \quad \forall n \in \mathbb{N}. \end{aligned}$$

Iterando en (3.39) obtenemos:

$$\sum_{q \in \Lambda_1^n} \|f_q\|^2 \leq \sum_{q \in \Lambda_1^{n-1}} \|f_q\|^2 \leq \dots \leq \sum_{q \in \Lambda_1^0} \|f_q\|^2 = \|U[e]f\|^2 = \|f\|^2, \quad \forall n \in \mathbb{N}. \quad (3.40)$$

La identidad (3.37) junto con las desigualdades (3.38) y (3.40) implican directamente que:

$$\|\Phi_{\Omega}^n(T_t f) - \Phi_{\Omega}^n(f)\|^2 \leq \frac{4\pi^2|t|^2K^2}{(S_1 \cdots S_n)^2} \|f\|^2, \quad (3.41)$$

para todo $n \in \mathbb{N}$. Falta probar (3.38). Para esto, primero vemos que:

$$\begin{aligned} & \|M_{-t/(S_1 \cdots S_n)}(\widehat{f_q * \chi_n}) - (\widehat{f_q * \chi_n})\|^2 \\ &= \int_{\mathbb{R}^d} \left| e^{-2\pi i \langle t, \omega \rangle / (S_1 \cdots S_n)} - 1 \right|^2 |\widehat{\chi_n}(\omega)|^2 |\widehat{f_q}(\omega)|^2 d\omega. \end{aligned} \quad (3.42)$$

Dado que $|e^{-2\pi i x} - 1| \leq 2\pi|x|$, para todo $x \in \mathbb{R}$, tenemos que:

$$|e^{-2\pi i \langle t, \omega \rangle / (S_1 \cdots S_n)} - 1|^2 \leq \frac{4\pi^2 |\langle t, \omega \rangle|^2}{(S_1 \cdots S_n)^2} \leq \frac{4\pi^2 |t|^2 |\omega|^2}{(S_1 \cdots S_n)^2}, \quad (3.43)$$

donde en el último paso aplicamos la *desigualdad de Cauchy-Schwarz*. Sustituyendo (3.43) en (3.42) obtenemos:

$$\begin{aligned} \|M_{-t/(S_1 \cdots S_n)}(\widehat{f_q * \chi_n}) - (\widehat{f_q * \chi_n})\|^2 &\leq \frac{4\pi^2 |t|^2}{(S_1 \cdots S_n)^2} \int_{\mathbb{R}^d} |\omega|^2 |\widehat{\chi_n}(\omega)|^2 |\widehat{f_q}(\omega)|^2 d\omega \\ &\leq \frac{4\pi^2 |t|^2 K^2}{(S_1 \cdots S_n)^2} \int_{\mathbb{R}^d} |\widehat{f_q}(\omega)|^2 d\omega, \end{aligned} \quad (3.44)$$

$$= \frac{4\pi^2 |t|^2 K^2}{(S_1 \cdots S_n)^2} \|\widehat{f_q}\|^2 = \frac{4\pi^2 |t|^2 K^2}{(S_1 \cdots S_n)^2} \|f_q\|^2, \quad (3.45)$$

para todo $n \in \mathbb{N}$, donde en (3.44) usamos la condición de decaimiento (3.30), y en el último paso, usamos de nuevo la fórmula de Parseval. Esto establece (3.38), y se completa así la prueba de (II).

□

Comenzamos notando que todas las no-linealidades puntuales $M_n : L^2(\mathbb{R}^d) \rightarrow L^2(\mathbb{R}^d)$ satisfacen la commutatividad en (3.28). Una gran clase de no-linealidades ampliamente utilizadas en la literatura de Aprendizaje Profundo, como las unidades lineales rectificadas, tangentes hiperbólicas, sigmoides logísticas desplazadas y la función módulo (empleada en el [Capítulo 2](#) al desarrollar la teoría de Mallat sobre redes de dispersión), son de hecho puntuales y, por lo tanto, cubiertas por el [Teorema 3.1](#). Además, $P = \text{Id}$ como en el submuestreo trivialmente satisface (3.28). El pooling promedio $Pf = f * \phi$, con $\phi \in L^1(\mathbb{R}^d) \cap L^2(\mathbb{R}^d)$, satisface (3.28) como consecuencia de que el operador de convolución commuta con el operador de traslación T_t .

3. Generalización de la Teoría de Mallat sobre Redes de Dispersión

Notamos que la condición (3.30) se puede cumplir fácilmente tomando los átomos generados de salida $\{\chi_n\}_{n \in \mathbb{N}_0}$ satisfaciendo:

$$\sup_{n \in \mathbb{N}_0} \{\|\chi_n\|_1 + \|Jac(\chi_n)\|_1\} < \infty.$$

La cota en (3.31) muestra que podemos controlar explícitamente la cantidad de invarianza por traslaciones mediante los factores de pooling S_n (cuanto mayores son los factores de pooling, mayor es la invarianza por traslaciones que se obtiene). Este resultado está en línea con observaciones hechas en la literatura de Aprendizaje Profundo, véase, por ejemplo, [HL06], [SWPo05], [MLo06], donde se argumenta informalmente que el pooling es crucial para obtener invarianza por traslaciones de las características extraídas. Además, la condición $\lim_{n \rightarrow \infty} S_1 \cdots S_n = \infty$ (fácilmente cumplida tomando $S_n > 1$, para todo $n \in \mathbb{N}$) garantiza, gracias a (3.31), una total invarianza por traslaciones asintótica pues:

$$\lim_{n \rightarrow \infty} \|\Phi_\Omega^n(T_t f) - \Phi_\Omega^n(f)\| = 0, \quad (3.46)$$

para todo $f \in L^2(\mathbb{R}^d)$ y $t \in \mathbb{R}^d$. Esto significa que las características $\Phi_\Omega^n(T_t f)$, correspondientes a las versiones desplazadas $T_t f$ del dígito manuscrito 3 en Figura 3.4 (b) y (c), se parezcan cada vez más a las características $\Phi_\Omega^n(f)$, correspondientes al dígito manuscrito 3 sin desplazar en Figura 3.4 (a), a medida que aumenta la profundidad de la red. En términos simples, el operador de desplazamiento T_t es progresivamente absorbido por Φ_Ω^n a medida que $n \rightarrow \infty$, con el límite superior (3.31) cuantificando esta absorción.

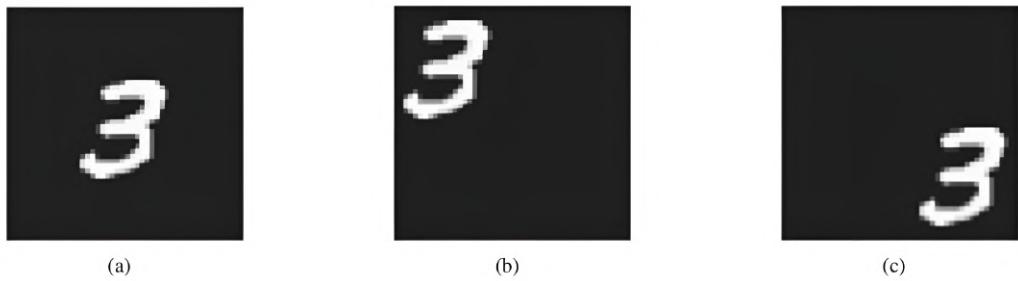


Figura 3.4.: Dígitos manuscritos del conjunto de datos MNIST [LC98]. Para tareas prácticas de Aprendizaje Automático (por ejemplo, clasificación de señales), a menudo deseamos que el vector de características $\Phi_\Omega(f)$ sea invariante a la ubicación espacial de los dígitos dentro de la imagen f . El Teorema 3.1 establece que las características $\Phi_\Omega^n(f)$ se vuelven más invariantes por traslaciones conforme aumenta el índice de la capa n . Imagen extraída de [WB18].

En contraste, el resultado de invarianza por traslaciones (3.4) desarrollado en la Sección 2.4 es asintótico en el parámetro de escala de ondícula J , y no depende de la profundidad de

la red, es decir, garantiza plena invarianza por traslaciones en cada capa de la red. Para diferenciar, nos referimos a (3.4) como invarianza horizontal por traslaciones y a (3.46) como invarianza vertical por traslaciones.

Destacamos que la invarianza vertical por traslaciones es una propiedad estructural. Específicamente, si P_n es unitario (como, por ejemplo, en el caso del submuestreo, donde P_n simplemente equivale a la función identidad), entonces también lo es la operación de pooling en (3.5) debido a:

$$\|S_n^{d/2}P_n(f)(S_n \cdot)\|^2 = S_n^d \int_{\mathbb{R}^d} |P_n(f)(S_n x)|^2 dx = \int_{\mathbb{R}^d} |P_n(f)(x)|^2 dx = \|P_n(f)\|^2 = \|f\|^2,$$

donde utilizamos el cambio de variable $y = S_n x$, $\frac{dy}{dx} = S_n^d$. En cuanto al pooling promedio, como se mencionó anteriormente, los operadores $P_n(f) = f * \phi_n$, $f \in L^2(\mathbb{R}^d)$, $n \in \mathbb{N}$, no son, en general, unitarios, pero aún así obtenemos invarianza por traslaciones como consecuencia de propiedades estructurales, a saber, la covarianza de traslación del operador de convolución combinada con la dilatación unitaria según (3.7).

Finalmente, señalamos que en la práctica, en ciertas aplicaciones, en realidad es la covarianza de traslación en el sentido de $\Phi_\Omega^n(T_t f) = T_t(\Phi_\Omega^n(f))$, para todo $f \in L^2(\mathbb{R}^d)$ y $t \in \mathbb{R}^d$, lo que es deseable, como por ejemplo, en la detección de landmarks faciales, donde el objetivo es estimar la posición absoluta de landmarks faciales en imágenes. En tales aplicaciones, las características en las capas más cercanas a la raíz de la red son más relevantes, ya que son menos invariantes a la traslación y más covariantes a la traslación. En [WTS⁺16] se proporciona evidencia numérica detallada de esto último. Presentamos a continuación la declaración formal de nuestra covarianza de traslación.

Corolario 3.1. *Sea $\Omega = ((\Psi_n, M_n, P_n))_{n \in \mathbb{N}}$ una secuencia de módulos admisible, sea $S_n \geq 1$, $n \in \mathbb{N}$, el factor de pooling en (3.10), y supongamos que los operadores $M_n : L^2(\mathbb{R}^d) \rightarrow L^2(\mathbb{R}^d)$ y $P_n : L^2(\mathbb{R}^d) \rightarrow L^2(\mathbb{R}^d)$ commutan con el operador de traslación T_t en el sentido de (3.28). Si, además, existe una constante $K > 0$ (que no depende de n) tal que las transformadas de Fourier $\widehat{\chi}_n$ de los átomos generadores de salida χ_n satisfacen la condición de decaimiento (3.30), entonces:*

$$\|\Phi_\Omega^n(T_t f) - T_t(\Phi_\Omega^n(f))\| \leq \frac{2\pi|t|K|1 - (S_1 \cdots S_n)|}{|S_1 \cdots S_n|} \|f\|,$$

para todo $f \in L^2(\mathbb{R}^d)$ y $t \in \mathbb{R}^d$.

Demostración. La idea principal de la demostración es (similar a la demostración de (II) en el Teorema 3.1) acotar la desviación de la covarianza perfecta en el dominio de la frecuencia. Para simplificar la notación, de nuevo, sea $f_q := U[q]f$, para $f \in L^2(\mathbb{R}^d)$. Gracias a (3.14) y la condición de admisibilidad (3.27), tenemos que $\|f_q\| \leq \|f\| < \infty$, y así $f_q \in L^2(\mathbb{R}^d)$. Primero escribimos:

3. Generalización de la Teoría de Mallat sobre Redes de Dispersión

$$\begin{aligned} \|\Phi_\Omega^n(T_t f) - T_t(\Phi_\Omega^n(f))\|^2 &= \|T_{t/(S_1 \dots S_n)} \Phi_\Omega^n(f) - T_t(\Phi_\Omega^n(f))\|^2 \\ &= \sum_{q \in \Lambda_1^n} \|(T_{t/(S_1 \dots S_n)} - T_t)(f_q * \chi_n)\|^2 \end{aligned} \quad (3.47)$$

$$= \sum_{q \in \Lambda_1^n} \|(M_{-t/(S_1 \dots S_n)} - M_{-t})(\widehat{f_q * \chi_n})\|^2, \quad (3.48)$$

para todo $n \in \mathbb{N}$, donde en (3.47) usamos (3.29), y en (3.48) empleamos la fórmula de Parseval (nótese que $(f_q * \chi_n) \in L^2(\mathbb{R}^d)$ gracias a la desigualdad de Young) junto con la relación:

$$\widehat{T_t f} = M_{-t} \widehat{f}, \quad \forall f \in L^2(\mathbb{R}^d), \forall t \in \mathbb{R}^d.$$

El paso clave es entonces establecer la siguiente cota superior:

$$\|(M_{-t/(S_1 \dots S_n)} - M_{-t})(\widehat{f_q * \chi_n})\|^2 \leq \frac{4\pi^2|t|^2 K^2 |1 - (S_1 \dots S_n)|^2}{|S_1 \dots S_n|^2} \|f_q\|^2, \quad (3.49)$$

donde $K > 0$ corresponde a la constante en la condición de decaimiento (3.30). Argumentos similares a los que conducen a (3.41) completan la prueba. Solo queda demostrar (3.49):

$$\begin{aligned} &\|(M_{-t/(S_1 \dots S_n)} - M_{-t})(\widehat{f_q * \chi_n})\|^2 \\ &= \int_{\mathbb{R}^d} \left| e^{-2\pi i \langle t, \omega \rangle / (S_1 \dots S_n)} - e^{-2\pi i \langle t, \omega \rangle} \right|^2 |\widehat{\chi_n}(\omega)|^2 |\widehat{f_q}(\omega)|^2 d\omega. \end{aligned} \quad (3.50)$$

Dado que $|e^{-2\pi i x} - e^{-2\pi i y}| \leq 2\pi|x - y|$, para todo $x, y \in \mathbb{R}$, se tiene que:

$$\left| e^{-2\pi i \langle t, \omega \rangle / (S_1 \dots S_n)} - e^{-2\pi i \langle t, \omega \rangle} \right|^2 \leq \frac{4\pi^2|t|^2 |\omega|^2 |1 - (S_1 \dots S_n)|^2}{|S_1 \dots S_n|^2}, \quad (3.51)$$

donde, nuevamente, empleamos la desigualdad de Cauchy-Schwarz. Sustituyendo (3.51) en (3.50), y empleando argumentos similares a los que conducen a (3.45), obtenemos (3.49), concluyendo así la demostración. \square

El [Corolario 3.1](#) muestra que, en ausencia de pooling, es decir, tomando $S_n = 1$, para todo $n \in \mathbb{N}$, se obtiene una covarianza de traslación completa en cada capa de la red. Esto demuestra que el pooling es necesario para lograr invarianza vertical por traslaciones, ya que de otro modo, las características permanecen completamente covariantes a las traslaciones,

independientemente de la profundidad de la red. Por otro lado, también observamos que, aunque el pooling contribuye a lograr invarianza por traslaciones, un aumento excesivo en los factores de pooling (S_1, \dots, S_n) no mejora la aproximación. De hecho, ocurre lo contrario: factores de pooling elevados reducen la resolución espacial, lo que puede degradar la precisión de la representación de las características. Finalmente, observamos que las redes de dispersión vistas en el [Capítulo 2](#) (que no emplean operadores de pooling) se vuelven invariantes horizontalmente por traslaciones al permitir que el parámetro de escala de la ondícula J tienda a infinito.

Para concluir, cabe destacar que en ([WB18], Sección IV-B), se demuestra también que el extractor de características convolucional general Φ_Ω es invariante frente a pequeñas deformaciones, es decir, se proporciona una cota de sensibilidad frente a deformaciones en señales de banda limitada $f \in L^2_R(\mathbb{R}^d) := \{f \in L^2(\mathbb{R}^d) \mid \text{supp}(\hat{f}) \subseteq B_R(0)\}$, siendo $B_R(0)$ la bola abierta de radio R centrada en 0, definidas como:

$$(T_{\tau, \omega} f)(x) := e^{2\pi i \omega(x)} f(x - \tau(x)),$$

donde se incluyen distorsiones no lineales y modulaciones no deseadas que afectan a la señal original (véase [Figura 3.5](#)). Debido a la complejidad técnica de esta sección y a la falta de tiempo, no se ha incluido en este trabajo.

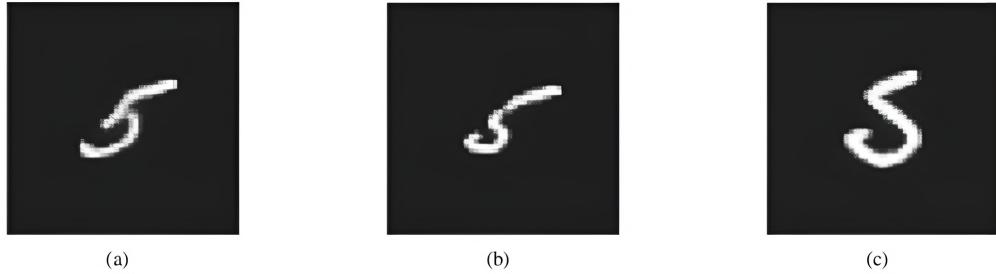


Figura 3.5.: Dígitos manuscritos del conjunto de datos MNIST [LC98]. Si f representa la imagen del dígito 5 en (a), entonces, con una función τ seleccionada apropiadamente, la función $T_\tau f = f(\cdot - \tau(\cdot))$ modela imágenes del 5 basadas en diferentes estilos de escritura como en (b) y (c). Imagen extraída de [WB18].

4. Conclusiones y Trabajos Futuros

En este trabajo, hemos abordado el estudio de las redes neuronales convolucionales desde una perspectiva teórica, centrándonos en su modelización matemática. A lo largo de los capítulos, hemos seguido un enfoque que busca fundamentar matemáticamente el funcionamiento de las redes neuronales convolucionales, tomando como punto de partida la teoría de Mallat y su aplicación en redes de dispersión.

En el primer capítulo, establecimos el marco teórico básico para entender la invarianza por traslaciones y la invarianza frente a pequeñas deformaciones. Esto nos llevó a considerar la necesidad de un operador que pudiera aplicarse de manera recursiva en las redes neuronales, manteniendo ciertas propiedades fundamentales.

En el segundo capítulo, se comenzó demostrando que los operadores derivados de la transformada de Fourier no son adecuados para nuestro propósito, debido a la falta de Lipschitz-continuidad bajo la acción de difeomorfismos. Por lo tanto, exploramos alternativas más viables, como las ondículas, especialmente la transformadas de Littlewood-Paley. Luego profundizamos en la teoría de Mallat sobre redes de dispersión, donde demostramos cómo las ondículas, combinadas con el operador módulo, pueden ser utilizadas para generar coeficientes invariantes por traslaciones. Se presentó un análisis riguroso sobre la propagación de la dispersión y la conservación de la norma en estas redes. Además, abordamos la estabilidad frente a pequeñas deformaciones y la importancia de la no-expansividad en la distancia entre funciones, destacando cómo las redes de dispersión mantienen la estabilidad bajo estas condiciones, lo que resulta esencial para aplicaciones de reconocimiento de patrones y clasificación.

Finalmente, en el tercer capítulo, extendimos la teoría de Mallat para cubrir arquitecturas convolucionales más generales. Aquí introdujimos un extractor de características convolucional que es capaz de capturar invarianza vertical por traslaciones gracias al uso de pooling en las distintas capas de la red, lo que refleja las observaciones experimentales en las aplicaciones prácticas de las redes neuronales convolucionales.

Con estas conclusiones, hemos logrado cumplir con los objetivos planteados al inicio del trabajo: proporcionar una modelización matemática general de las redes neuronales convolucionales y demostrar la invarianza por traslaciones en un marco teórico sólido. Respecto a la invarianza frente a pequeñas deformaciones, sí se demostró para redes de dispersión, pero no se ha hecho formalmente para la modelización general. Este recorrido, además de haber permitido un análisis profundo de las redes neuronales convolucionales desde un enfoque matemático, me ha proporcionado una mejor comprensión de conceptos fundamentales como las transformadas de ondículas, los marcos semi-discretos, etc.

Este trabajo también ha sido una oportunidad para profundizar en áreas clave del Procesamiento de Señales y la Visión por Computador, conectando de manera natural los conocimientos adquiridos durante los dos grados que he cursado. El proceso de investigación

4. Conclusiones y Trabajos Futuros

me ha permitido aprender a leer y analizar publicaciones matemáticas especializadas, una habilidad que considero esencial para futuras investigaciones en este campo.

Teniendo en cuenta los resultados obtenidos en este trabajo, los trabajos futuros podrían ser:

- Demostrar rigurosamente que $(\bar{S}f)(p)$, $\forall f \in L^2(\mathbb{R}^d)$, y para todo camino p , pertenece a $L^2(\mathbb{R}^d)$.
- Demostrar la condición suficiente que garantiza la convergencia uniforme del operador de dispersión de ventana $S_J f$ en $\bar{S}f$.
- Demostrar que el propagador dispersa la energía progresivamente hacia bajas frecuencias ([Lema 2.4](#)).
- Demostrar la estabilidad de la transformada de ondículas W_J frente a pequeñas deformaciones ([Lema 2.9](#)).
- Demostrar la invarianza frente a pequeñas deformaciones del extracto de características convolucionales general.

Parte II.

Técnicas de Explicabilidad para Arquitecturas Neuronales Profundas de Segmentación de Instancias

5. Introducción

5.1. Descripción del problema y motivación

En el campo de la Visión por Computador, la segmentación de instancias es una tarea fundamental que permite identificar y aislar cada objeto en una imagen, proporcionando una máscara específica para cada instancia detectada. Este proceso combina la detección de objetos y la segmentación semántica (véase [Figura 5.1](#)), ya que no solo busca identificar la presencia de un objeto en una imagen, sino también delinear con precisión sus contornos. Una de las arquitecturas más destacadas en esta tarea es Mask R-CNN [[HGDG17](#)], que extiende la estructura de Faster R-CNN [[RHGS15](#)] para lograr una segmentación precisa de instancias. A pesar de su eficacia, Mask R-CNN y otras redes neuronales profundas son vistas como cajas negras, lo que dificulta comprender y justificar sus decisiones, un aspecto crítico en aplicaciones que requieren transparencia, como la medicina, la conducción autónoma y la toma de decisiones éticas.

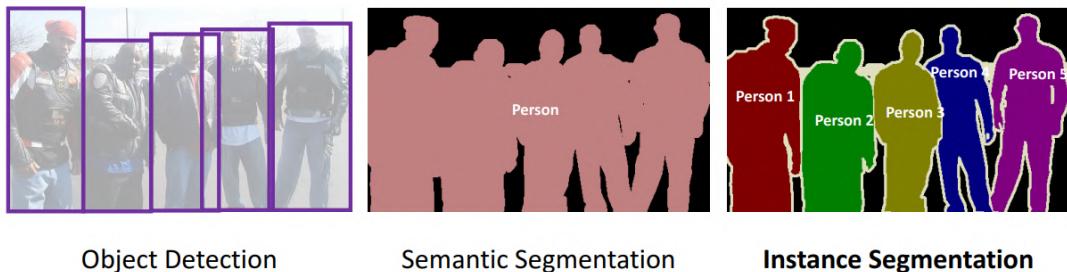


Figura 5.1.: La segmentación de instancias combina la detección de objetos y la segmentación semántica. Imagen extraída de [[Liu18](#)].

Este trabajo se centra en abordar el problema de la interpretabilidad de Mask R-CNN mediante la implementación de técnicas de explicabilidad. En particular, se aplicará Grad-CAM [[SCD⁺17](#)] a dicha arquitectura, un método de interpretabilidad post-hoc que genera mapas de calor para visualizar las regiones más influyentes de la imagen en las predicciones de la red (véase [Figura 5.2](#)). Esta técnica no solo facilita la comprensión del modelo, sino que también aporta explicaciones visuales de sus decisiones, permitiendo al usuario identificar las áreas de la imagen que guían las clasificaciones y segmentaciones de Mask R-CNN, contribuyendo así al avance de la Inteligencia Artificial Explicable.

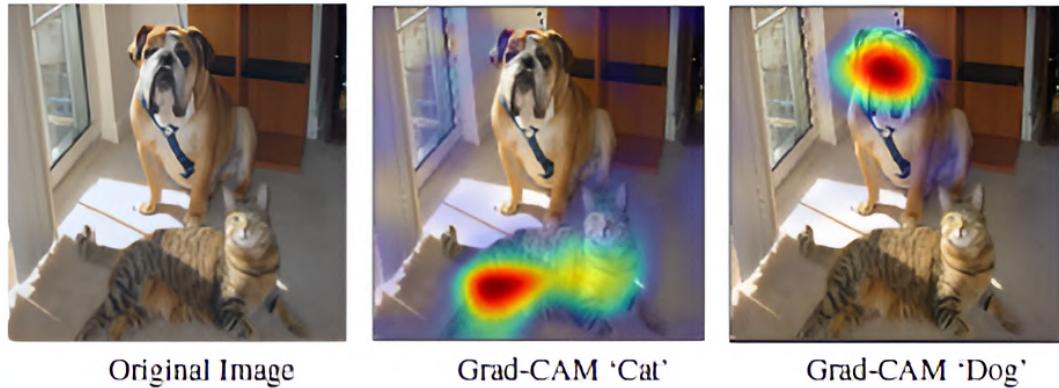


Figura 5.2.: Visualización de mapas de calor generados por Grad-CAM para una imagen de entrada que contiene un perro y un gato. La imagen central muestra el mapa de calor correspondiente a la predicción de la clase *Cat*, destacando las áreas relevantes para dicha clasificación. La imagen de la derecha presenta el mapa de calor para la clase *Dog*, resaltando las zonas de la imagen que la red considera importantes para identificar al perro. Imagen extraída de [SCD⁺₁₇].

5.2. Objetivos

Para alcanzar los fines propuestos en este trabajo, se han definido los siguientes objetivos específicos:

1. **Estudiar la segmentación de instancias y el funcionamiento de Mask R-CNN:** Comprender la arquitectura de Mask R-CNN, desde su estructura de red hasta los distintos componentes que permiten la detección y segmentación de instancias, sentando una base sólida para su aplicación en este trabajo.
2. **Investigar el campo de la Inteligencia Artificial Explicable:** Explorar los métodos de interpretabilidad, especialmente Grad-CAM, que permite visualizar las áreas relevantes de una imagen para las predicciones del modelo. Este objetivo implica analizar en profundidad cómo funciona Grad-CAM y su aplicación en redes neuronales convolucionales.
3. **Implementar una adaptación de Grad-CAM en Mask R-CNN:** Aplicar y adaptar Grad-CAM en la arquitectura de Mask R-CNN utilizando bibliotecas de explicabilidad como InterpretDL, con el objetivo de dotar de interpretabilidad a las predicciones realizadas por el modelo en el contexto de la segmentación de instancias. Dicha implementación se basará en adaptaciones propuestas en investigaciones previas [XJ₂₁], [XVM⁺₂₁], y [XVM⁺₂₂].
4. **Evaluar la efectividad de Grad-CAM en Mask R-CNN:** Analizar el rendimiento y la utilidad de Grad-CAM aplicado a Mask R-CNN mediante pruebas experimentales,

5.3. Planificación y estimación de costes

evaluando la calidad de las explicaciones generadas y determinando el grado en que facilitan la interpretabilidad de la segmentación de instancias.

5.3. Planificación y estimación de costes

HACER ESTA SECCIÓN 5.3 AL FINAL.

6. Fundamentos Teóricos

En este capítulo se presentarán los conceptos teóricos esenciales que sirven de base para este trabajo y sus conclusiones. Para ello, se ha utilizado el conocimiento obtenido en asignaturas como Inteligencia Artificial, Visión por Computador y Aprendizaje Automático, además de varios artículos especializados que se mencionarán en los puntos correspondientes. Asimismo, parte de la información sobre los conceptos ha sido extraída del curso de Stanford CS231n [Unizo].

6.1. Aprendizaje Automático

Hoy en día, la Inteligencia Artificial (*Artificial Intelligence* en inglés) ha ganado una importancia considerable dentro del campo de la informática [RNo2]. Su objetivo principal es proporcionar a los ordenadores la capacidad de razonar y resolver problemas de manera autónoma e inteligente. Dentro de este amplio ámbito de estudio, se han desarrollado diversas estrategias para alcanzar dichos objetivos, destacando áreas como las Metaheurísticas y, de manera especial, el Aprendizaje Automático (*Machine Learning* en inglés). En este trabajo, nos centramos en el Aprendizaje Automático, por lo que es fundamental definir este concepto con claridad.

Una de las primeras definiciones proviene de Arthur Samuel, quien en 1959 definió el Aprendizaje Automático como la capacidad de los ordenadores para aprender sin necesidad de una programación explícita [Sam59]. Si bien esta definición puede parecer algo vaga, ofrece una visión general del propósito del Aprendizaje Automático: permitir que las máquinas aprendan a resolver problemas a partir de datos de entrenamiento sin instrucciones directas.

Tom Mitchell, en 1997, refinó esta definición al establecer que un programa se considera que aprende cuando, al realizar una tarea T y utilizando una medida de rendimiento P , mejora su desempeño gracias a la experiencia E acumulada en la tarea [Mit97]. Esta versión nos proporciona una estructura más clara para entender cómo abordar problemas mediante técnicas de Aprendizaje Automático.

En resumen, los elementos clave del Aprendizaje Automático son un problema T que queremos resolver, la experiencia E , que normalmente se refiere a un conjunto de datos asociados, y una medida de rendimiento P que evalúa cómo mejora el algoritmo a medida que se entrena. Generalmente, esta medida de rendimiento está vinculada a una función objetivo que buscamos optimizar.

Los algoritmos de Aprendizaje Automático se dividen comúnmente en dos grandes categorías dependiendo de la naturaleza de los datos de entrenamiento: el Aprendizaje Supervisado y el Aprendizaje No Supervisado. Además, en los últimos años ha ganado relevancia el Aprendizaje por Refuerzo, aunque no profundizaremos en esta técnica, ya que no es esencial

6. Fundamentos Teóricos

para el enfoque de este trabajo.

6.1.1. Aprendizaje Supervisado

Los algoritmos de Aprendizaje Automático dentro de esta categoría se caracterizan por contar con un conjunto de datos en los que cada muestra x tiene una etiqueta asociada y . El objetivo principal es encontrar la función f que relaciona x con y , de modo que $f(x) = y$.

6.1.1.1. Problemas de regresión

En los problemas de regresión, buscamos determinar la función f que asocia cada muestra con su etiqueta de manera adecuada:

$$f(x) = y, \quad \text{donde } x \in \mathbb{R}^m, \quad y \in \mathbb{R}^n,$$

Dado que encontrar la función exacta f suele ser difícil, tratamos de aproximarla con una función \hat{f} , que normalmente pertenece a una familia de funciones parametrizadas. Esta función será entrenada utilizando los datos etiquetados proporcionados. Siguiendo la definición de Tom Mitchell, los elementos clave en este tipo de problemas serían:

- T = Regresión (aproximación de f).
- E = Conjunto de datos etiquetados X que se utiliza para entrenar el modelo \hat{f} .
- P = Función de coste (usualmente el error cuadrático medio) que evalúa qué tan bien \hat{f} se approxima a f .

Como ejemplo, consideremos la formalización de un caso práctico en el que queremos predecir f utilizando un modelo lineal. En este caso, la función \hat{f} tendría la siguiente forma:

$$\hat{f}(x, w) = w^T x, \quad x, w \in \mathbb{R}^m,$$

donde w es el conjunto de parámetros que ajustaremos para aproximar de la mejor manera la función f . Además, contamos con un conjunto de N ejemplos de datos:

$$X = \{x_1, x_2, \dots, x_N\}, \quad x_i \in \mathbb{R}^m,$$

y sus respectivas etiquetas:

$$Y = \{y_1, y_2, \dots, y_N\}, \quad y_i \in \mathbb{R}^n,$$

Consideremos que, para este ejemplo, usamos como función de coste \mathcal{L} el error cuadrático medio, que es comúnmente utilizado en este tipo de aproximaciones. La función de coste sería:

$$\mathcal{L}(w) = \frac{1}{N} \sum_{i=1}^N \left(\hat{f}(x_i, w) - f(x_i) \right)^2 = \frac{1}{N} \sum_{i=1}^N (\hat{y}_i - y_i)^2,$$

donde \hat{y}_i representa la etiqueta predicha por \hat{f} para la entrada x_i .

Con estos elementos, el objetivo es encontrar el vector de pesos w que minimice la función de coste \mathcal{L} , utilizando los datos de entrenamiento X para ello.

6.1.1.2. Problemas de clasificación

Además, encontramos los problemas de clasificación, donde los datos están organizados en distintas clases y el objetivo es asignar cada entrada a la clase correspondiente. Uno de los problemas más comunes en este ámbito es el de la clasificación binaria, donde se intenta predecir si un dato pertenece o no a una clase particular en un conjunto, codificado como 0 o 1.

En estos problemas, se busca una forma de definir la frontera más precisa posible entre los distintos tipos de datos que deseamos separar. Algunos de los algoritmos más utilizados en esta tarea son los siguientes:

- **K-nearest neighbours:** Este algoritmo asigna a cada dato la etiqueta correspondiente según los K vecinos más cercanos (donde $K \in \mathbb{N}$) del conjunto [Koz08].
- **Máquinas de vectores de soporte:** Se centra en hallar el hiperplano que maximiza la distancia entre los puntos de los distintos conjuntos, lo que se conoce como margen [HDO⁺98].
- **Redes neuronales:** En futuras secciones se profundizará en estas, destacando el perceptrón multicapa [Bis94].

6.1.1.3. Gradiente descendente

Hasta este punto hemos hablado sobre qué es el Aprendizaje Automático y cómo se formulan sus problemas. Sin embargo, aún no hemos discutido ningún algoritmo que se utilice para minimizar una función de coste. Por ello, a continuación, explicaremos un algoritmo que es

6. Fundamentos Teóricos

tradicionalmente empleado para esta tarea: el gradiente descendente [Rud16].

El gradiente descendente es un método clásico que sigue la idea intuitiva de que el gradiente de una función siempre indica la dirección del máximo de la función. Siguiendo la dirección opuesta, se puede alcanzar el mínimo de la función.

Siguiendo la notación del apartado anterior, si la función objetivo es:

$$f(x) = y, \quad x \in \mathbb{R}^m, \quad y \in \mathbb{R}^n,$$

y la función que se va a utilizar para aproximar la función objetivo es:

$$\hat{f}(x, w) = w^T x, \quad x, w \in \mathbb{R}^m,$$

La función de coste $\mathcal{L}(w)$ mide de alguna forma la diferencia entre f y \hat{f} , y es necesario que sea derivable para poder aplicar este método. Algunas de las funciones de coste más comunes son:

- **Función L2 (conocida como error cuadrático medio):**

$$\mathcal{L}(w) = \frac{1}{N} \sum_{i=1}^N (\hat{f}(x_i, w) - f(x_i))^2$$

- **Función L1 (conocida como error absoluto medio):**

$$\mathcal{L}(w) = \frac{1}{N} \sum_{i=1}^N |\hat{f}(x_i, w) - f(x_i)|$$

Es importante tener en cuenta que la función de coste puede variar considerablemente de un problema a otro, y en algunos casos puede ser una combinación de varias funciones o una función única diseñada para un problema específico.

Una vez que se ha formalizado el problema, el algoritmo de gradiente descendente se implementa de la siguiente manera:

- Inicializamos el vector de pesos w siguiendo un criterio determinado.

- En cada iteración i del entrenamiento, el vector de pesos en la siguiente iteración se actualiza según la relación:

$$w_{i+1} = w_i - \eta \nabla \mathcal{L}(w),$$

donde η es el parámetro conocido como la tasa de aprendizaje (*learning rate* en inglés), que determina el tamaño del paso que damos en cada iteración en la búsqueda del mínimo, y $\nabla \mathcal{L}(w)$ es el gradiente de la función de coste respecto al vector de pesos w .

Este método, en su forma ideal, debería encontrar el mínimo global de la función de coste siempre que esta sea convexa. Sin embargo, en caso de no serlo, es posible que el algoritmo converja a un mínimo local.

Además, es fundamental elegir adecuadamente la tasa de aprendizaje. Si η es demasiado pequeño, el algoritmo puede converger muy lentamente hacia el mínimo, lo que implicaría realizar un número elevado de iteraciones. Por el contrario, si η es demasiado grande, el algoritmo puede no converger en absoluto, ya que los saltos en la dirección del mínimo podrían ser demasiado grandes y sobrepasar el mínimo en cada iteración. Una estrategia habitual para abordar este problema es utilizar una tasa de aprendizaje adaptativa, que comienza siendo grande y se reduce progresivamente a medida que se incrementa el número de iteraciones, lo que permite acercarse al mínimo de manera más controlada (véase Figura 6.1).

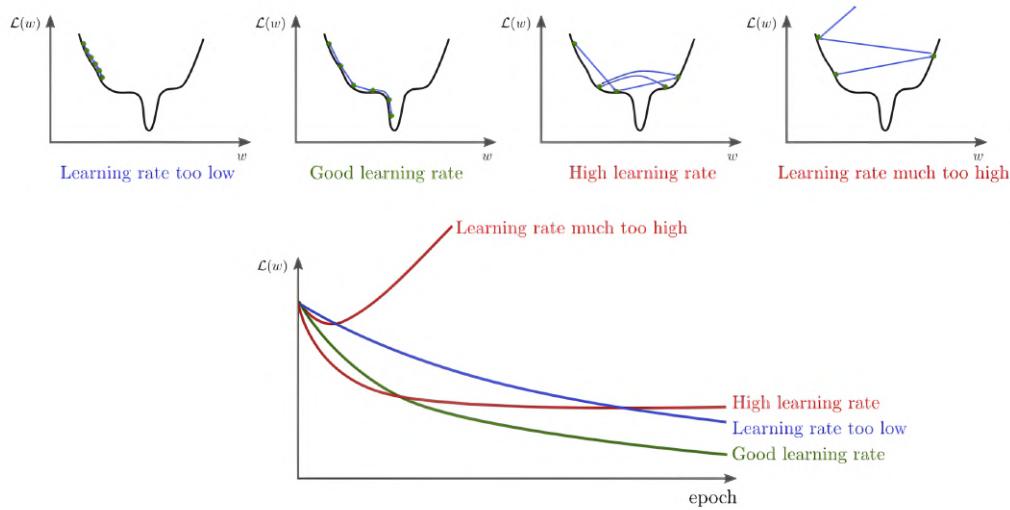


Figura 6.1.: Importancia de una elección adecuada de la tasa de aprendizaje. Imagen extraída de [Uni20].

6. Fundamentos Teóricos

6.1.1.4. Gradiente descendente estocástico

El algoritmo de gradiente descendente descrito anteriormente presenta el inconveniente de ser costoso desde el punto de vista computacional, ya que en cada iteración se debe calcular la función de coste para todos los ejemplos del conjunto de entrenamiento X . Para aliviar esta carga, se emplea una variante que sigue ofreciendo buenos resultados. Esta variante consiste en actualizar los pesos usando solo una pequeña parte de los ejemplos del conjunto de entrenamiento X (este pequeño conjunto de ejemplos de entrenamiento se conoce como *mini-batch*).

6.1.2. Aprendizaje No Supervisado

Los algoritmos de Aprendizaje No Supervisado se distinguen por trabajar con datos que no tienen etiquetas, es decir, no se dispone de salidas conocidas. En lugar de buscar una respuesta concreta, el objetivo es analizar el conjunto de datos y extraer patrones o características relevantes. Por ejemplo, un uso común de este tipo de algoritmos es el agrupamiento de clientes de una empresa en diferentes categorías según sus características.

6.2. Aprendizaje Profundo

Curiosamente, algunas de las tareas que resultan sencillas para los humanos, como el reconocimiento del habla o la identificación de objetos en imágenes, han representado un desafío considerable para los ordenadores. Sin embargo, en los últimos años, con la llegada del Aprendizaje Profundo (*Deep Learning* en inglés) [LBH15], se han logrado avances significativos en estos campos, obteniendo resultados cada vez más satisfactorios.

Los algoritmos de Aprendizaje Profundo abordan estos problemas descomponiéndolos en múltiples capas de representaciones que se construyen a partir de otras más simples. Así, conceptos complejos pueden derivarse de otros más básicos. Dado que este tipo de modelo puede llegar a tener muchas capas, es por ello que se le denomina Aprendizaje Profundo.

6.2.1. Redes neuronales

La estructura básica de los modelos de Aprendizaje Profundo está basada en las redes neuronales. Por esta razón, profundizaremos en esta arquitectura, utilizando como ejemplo el perceptrón multicapa.

Como mencionamos anteriormente, el objetivo de las redes neuronales es aproximar una función desconocida $f(x) = y$ que, dada una entrada x , genera una salida y . Esto se logra a través de una función $\hat{f}(x, w) = \hat{y}$, donde \hat{y} es la salida predicha para la entrada x , y w es el conjunto de parámetros que ajustaremos para que la red neuronal pueda aproximar de la mejor manera la función objetivo f .

Los algoritmos utilizados por las redes neuronales son de tipo feedforward, ya que la información fluye desde la entrada x a través de múltiples cálculos, hasta llegar a la salida

\hat{y} . Debido a la forma en que se organizan estos cálculos, estos algoritmos se conocen como redes, ya que están formados por una serie de funciones aplicadas de manera secuencial. La profundidad de una red depende del número de capas ocultas que contiene.

Adicionalmente, cada capa está compuesta por varias neuronas, que se conectan tanto a las neuronas de la capa anterior como a las de la capa siguiente a través de combinaciones lineales (véase Figura 6.2). Sin embargo, las combinaciones lineales por sí solas no son suficientes para que la red sea capaz de aproximar funciones complejas no-lineales, por lo que se emplean las llamadas funciones de activación. Estas funciones no-lineales son esenciales, y entre las más utilizadas se encuentran las siguientes (véase Figura 6.3):

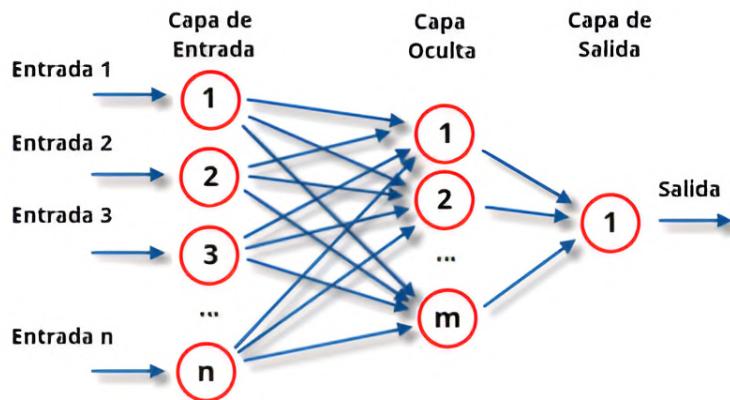


Figura 6.2.: Perceptrón multicapa con una capa oculta. Formalmente, se puede expresar como $\hat{f}(x, w) = f_2(f_1(x, w))$, donde f_1 representa la transformación de los datos de la capa de entrada a la capa oculta, y f_2 es la transformación de la capa oculta a la capa de salida. Imagen extraída de [Wik24c].

- **Función Sigmoide:** Convierte los valores de entrada a un rango entre 0 y 1.

$$\text{sigmoide}(x) = \frac{1}{1 + e^{-x}}.$$

- **Función Tangente Hiperbólica:** Similar a la sigmoide, pero simétrica con respecto al origen.

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}.$$

- **Función ReLU:** Es una de las funciones más populares en redes neuronales por su eficiencia, ya que permite que solo se activen ciertas neuronas. Las que no son activadas

6. Fundamentos Teóricos

tras la combinación lineal de la capa oculta simplemente no se utilizan.

$$ReLU(x) = \max(0, x).$$

Sin embargo, existe el inconveniente de que el gradiente puede ser 0 en algunos casos, lo que provoca que los pesos no se actualicen, un problema que se abordará más adelante cuando hablemos de back-propagation.

- **Función Leaky ReLU:** Una mejora de la ReLU que permite evitar que el gradiente sea cero cuando la función es negativa, añadiendo un pequeño valor lineal para x negativo.

$$\text{LeakyReLU}(x) = \begin{cases} x & \text{si } x \leq 0 \\ \alpha x & \text{si } x > 0 \end{cases},$$

siendo α un número muy cercano a 0.

- **Función Softmax:** Se trata de una combinación de múltiples funciones sigmoide. Mientras que la función sigmoide es más comúnmente utilizada en problemas de clasificación binaria, la función softmax permite realizar clasificación en múltiples clases. Esto se logra transformando un vector de entrada de K dimensiones, compuesto por valores reales, en un vector de salida también de K dimensiones, donde cada valor está entre 0 y 1. En este contexto, la clase correspondiente será aquella cuya componente sea más cercana a 1, lo que permite identificar la clase a la que pertenece el elemento en un problema de clasificación multiclasa.

$$\text{softmax}(x)_j = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}}, \quad j = 1, \dots, K.$$

No existe un método único para seleccionar la función de activación óptima en cada situación, pero los resultados experimentales han mostrado que la función ReLU tiende a proporcionar buenos resultados. Si en algún momento se presentan demasiadas neuronas inactivas, es posible cambiar a la función Leaky ReLU para mejorar el rendimiento.

De esta forma, la función f_i en la capa i de la red suele expresarse de la siguiente manera:

$$f_i(x, w) = \gamma(w_i^T x),$$

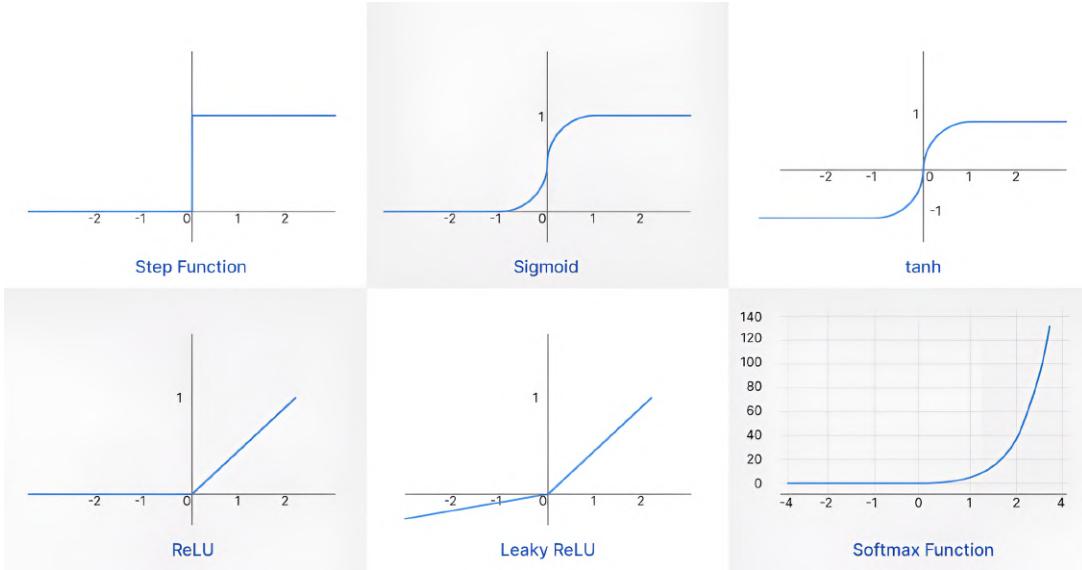


Figura 6.3.: Gráficas de las funciones de activación más utilizadas. Imagen extraída de [\[Onl24\]](#).

donde γ representa una función de activación, la cual puede ser cualquiera de las funciones previamente mencionadas, y w_i es el vector de pesos asociado a la capa i -ésima de la red. Esto permite que la red sea capaz de aproximar funciones no lineales de forma eficiente.

6.2.2. Back-propagation

Todas las funciones en las capas intermedias de una red neuronal son derivables, lo que implica que en teoría se podría calcular su derivada de manera explícita. Sin embargo, este proceso puede ser computacionalmente costoso. En lugar de hacerlo directamente, se utiliza el método de back-propagation.

Para comprender mejor este proceso, hacemos uso del concepto de grafo computacional, que no es más que una representación gráfica de una función.

La idea detrás del algoritmo de back-propagation es calcular las derivadas en cada nodo del grafo computacional usando la regla de la cadena. Por ejemplo, si tenemos una función $f(x, y, z) = g(x, y)h(z)$ y queremos calcular $\frac{\partial f}{\partial x}$, aplicando la regla de la cadena obtenemos:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial g} \frac{\partial g}{\partial x}.$$

En este método, el flujo de la información va desde la salida hacia la entrada, calculando los gradientes para cada parámetro a lo largo del camino. Una vez que se obtiene el gradiente,

6. Fundamentos Teóricos

se procede a actualizar los pesos, utilizando métodos como el algoritmo de gradiente descendente.

Para comprender mejor este procedimiento, véase [Figura 6.4](#).

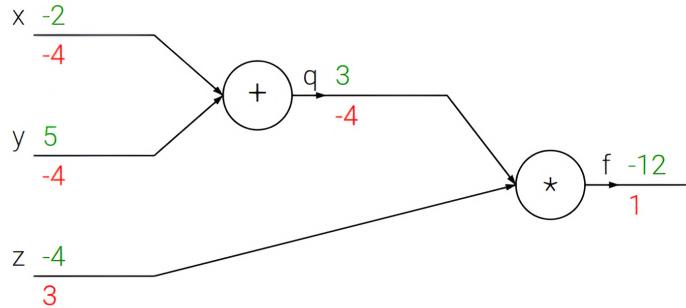


Figura 6.4.: Ejemplo de back-propagation representando la función $f(x,y,z) = (x+y)z$, para valores $x = -2$, $y = 5$, $z = -4$, en un grafo computacional. El grafo muestra el flujo de cálculo de la salida $f = -12$. Primero se calculan los valores desde las entradas hasta la salida (mostrados en verde). Luego, se realiza el back-propagation, que comienza al final y aplica recursivamente la regla de la cadena para calcular los gradientes (mostrados en rojo) hasta llegar a las entradas del circuito. Imagen extraída de [\[Unizo\]](#).

6.3. Redes neuronales convolucionales

Las redes neuronales convolucionales son una de las herramientas más importantes en el campo del Aprendizaje Profundo. Estas redes están inspiradas en las redes neuronales previamente descritas, pero con la diferencia de que, en lugar de trabajar con entradas de dimensión \mathbb{R}^d , ahora pueden procesar datos volumétricos en múltiples dimensiones, como imágenes representadas en el espacio $\mathbb{R}^{w \times h \times c}$.

El concepto clave detrás de estas redes es el de conectividad local, que permite trabajar eficientemente con entradas de grandes dimensiones, como imágenes. Conectar cada neurona con todas las de la capa anterior sería impráctico en estos casos, como ocurre en las redes neuronales clásicas vistas anteriormente. En este contexto, conectamos cada neurona a una región local del volumen de entrada, y utilizamos filtros que mantienen la misma profundidad que dicho volumen, los cuales desplazamos a lo largo de $\mathbb{R}^{w \times h \times c}$. La operación realizada por estos filtros se describe a continuación, y su resultado alimenta a la siguiente capa de la red.

Estas redes se denominan convolucionales porque la operación matemática clave que llevan a cabo es la convolución. La convolución entre dos funciones de una variable real se define

mediante la siguiente ecuación:

$$f(t) = (g * h)(t) = \int g(a)h(t - a)da,$$

donde t usualmente representa el tiempo, mientras que g y h son funciones de una variable real.

Sin embargo, la expresión anterior corresponde a una definición en el caso continuo. Al trabajar en un entorno computacional, debemos discretizar tanto el tiempo como las funciones utilizadas. Por lo tanto, asumimos que t toma valores enteros. De esta manera, la convolución discreta en una dimensión se define como:

$$f(t) = (g * h)(t) = \sum_{a=-\infty}^{\infty} g(a)h(t - a).$$

No obstante, en el caso de redes neuronales convolucionales aplicadas a imágenes, que se representan como matrices 2D, necesitamos extender esta definición al caso bidimensional. Así, la convolución discreta en dos dimensiones se expresa como:

$$f(n, m) = (g * h)(n, m) = \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} g(i, j)h(n - i, m - j).$$

La operación de convolución posee las siguientes propiedades:

- **Commutatividad:**

$$f * g = g * f$$

- **Asociatividad:**

$$f * (g * h) = (f * g) * h$$

- **Distributividad:**

$$f * (g + h) = (f * g) + (f * h)$$

Con estos conceptos establecidos, podemos empezar a describir la estructura de una red neuronal convolucional, la cual está compuesta principalmente por tres tipos de capas: las capas convolucionales, las capas de pooling y las capas totalmente conectadas (véase [Figura 6.5](#) y [Figura 6.6](#)).

6. Fundamentos Teóricos

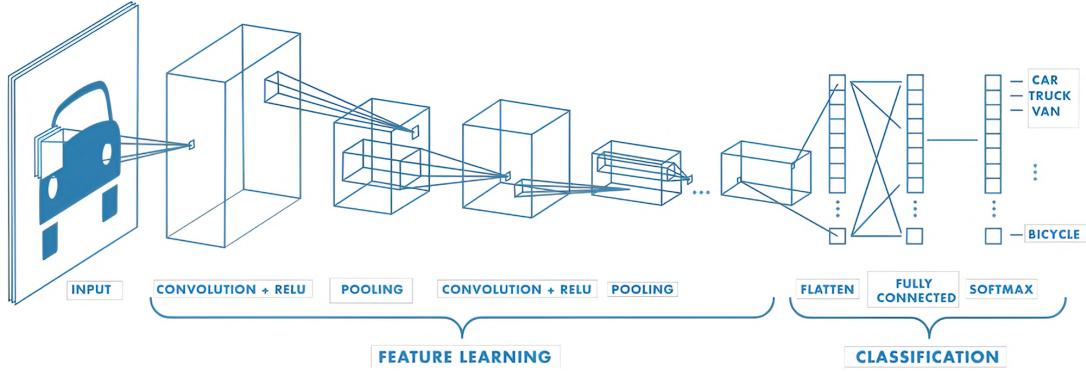


Figura 6.5.: Diagrama esquemático de la arquitectura de una red neuronal convolucional, que ilustra el proceso de aprendizaje de características y la etapa de clasificación. Las capas de convolución y pooling permiten la extracción progresiva de características significativas, mientras que las capas finales totalmente conectadas realizan la clasificación en categorías predefinidas, como coche o bicicleta. Imagen extraída de [Mat24b].

6.3.1. Capa convolucional

Una capa clave en las redes neuronales convolucionales es la capa convolucional, que se basa en mover un filtro a lo largo del volumen de entrada y realizar una operación de convolución discreta en dos dimensiones. Como mencionamos anteriormente, la profundidad del filtro siempre coincide con la del volumen de entrada. Ahora exploraremos cómo se determina el volumen de salida generado por esta capa, que depende de tres parámetros principales: depth, stride y padding.

- **Parámetro depth:** Se refiere al número de filtros que aplicamos al volumen de entrada, donde cada filtro aprenderá a detectar diferentes características. Cada filtro genera un volumen de tamaño $m \times n$ y profundidad 1, conocido como el mapa de activación, y este parámetro define cuántos mapas de activación se generan (véase Figura 6.7).
- **Parámetro stride:** Define el desplazamiento del filtro sobre el volumen de entrada. Un stride de 1 significa que el filtro se desplaza de uno en uno sobre los píxeles, mientras que un stride de 2 indica que el filtro se mueve de dos en dos. Cuanto mayor sea el valor del stride, menor será la dimensión del mapa de activación resultante.
- **Parámetro padding:** Indica cuántas filas y columnas se agregan alrededor del volumen de entrada antes de aplicar los filtros. Este parámetro permite controlar la dimensión de los mapas de activación, ya que la operación de convolución tiende a reducir la dimensionalidad, a menos que el filtro utilizado sea de tamaño 1×1 .

Considerando todos estos parámetros, la fórmula que calcula el volumen de salida de una capa convolucional, cuando el volumen de entrada tiene dimensiones $W \times W \times d$ y se em-

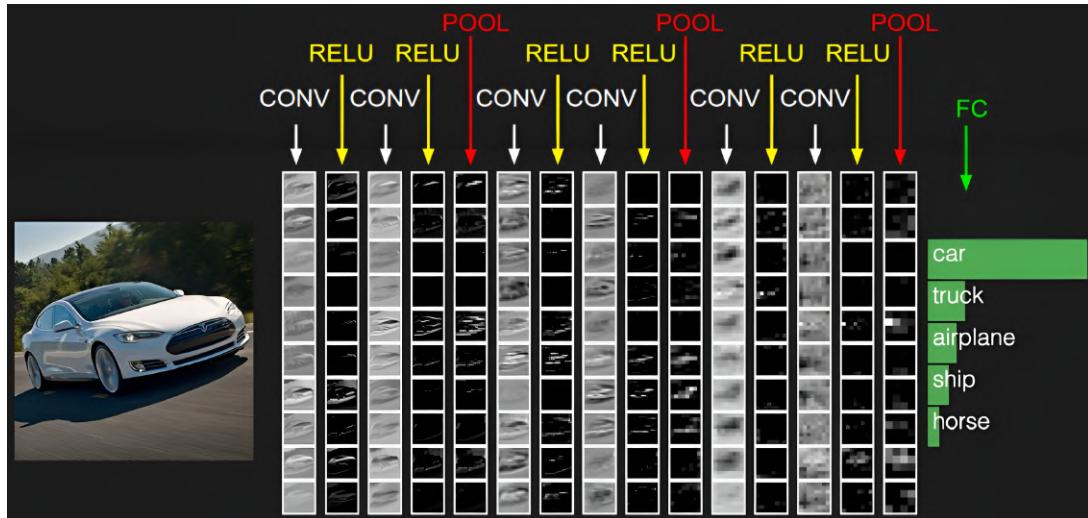


Figura 6.6.: Ejemplo visual del flujo de datos en una red neuronal convolucional, donde se observa cómo las capas de convolución (CONV) y activación (ReLU) extraen características de la imagen de entrada, seguidas de capas de pooling (POOL) que reducen la dimensión espacial manteniendo la profundidad. Al final, una capa totalmente conectada (FC) clasifica la imagen en una categoría específica, siendo coche en este caso. Imagen extraída de [Unizo].

plean filtros de tamaño $F \times F \times d$, con un padding P y un stride S , es la siguiente:

$$\text{Output} = \frac{(W - F + 2P)}{S} + 1$$

En general, la salida de una capa convolucional se pasa por una función de activación (véase Figura 6.8). Por lo general, se utiliza la función ReLU, aunque es común también emplear la Leaky ReLU.

6.3.2. Capa de pooling

En las redes neuronales convolucionales, es común introducir capas de pooling de manera periódica. Esta capa tiene la función de reducir las dimensiones del volumen de entrada, actuando de forma independiente de la profundidad del mismo. Normalmente, se utilizan filtros de tamaño 2×2 con un stride de 2, lo que reduce las dimensiones del volumen de entrada a la mitad, manteniendo la profundidad inalterada.

Existen varias operaciones que se pueden realizar con el filtro 2×2 , las cuales han sido objeto de investigación en los últimos años. Entre ellas, destacan las siguientes:

- **Max-pooling:** Esta operación selecciona el valor máximo de los cuatro elementos que

6. Fundamentos Teóricos

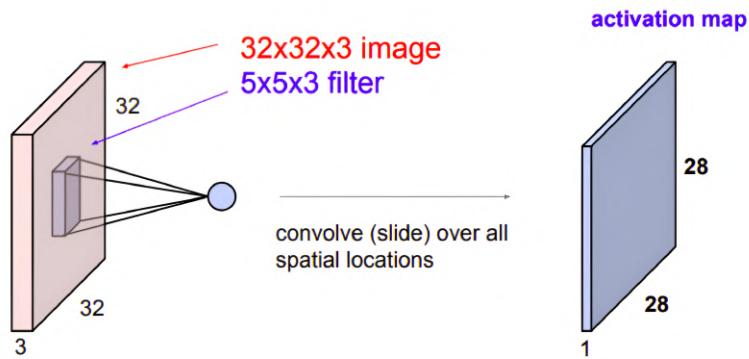


Figura 6.7.: Ejemplo que ilustra cómo se calcula un mapa de activación en una capa convolucional para un volumen de entrada específico. Imagen extraída de [Unizo].

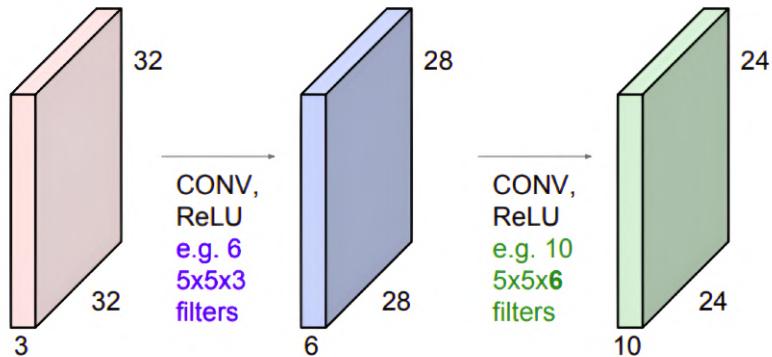


Figura 6.8.: Secuencia de varias capas convolucionales seguidas de la función de activación. Es importante señalar que la profundidad de los filtros siempre coincide con la del volumen de entrada. Imagen extraída de [Unizo].

cubre el filtro en el volumen de entrada.

- **Average pooling:** En esta operación, se calcula el promedio de los elementos que abarca el filtro.

En general, el max-pooling muestra mejor rendimiento en la práctica [SHII20], aunque este tema sigue siendo investigado.

6.3.3. Capa totalmente conectada

Al final de las redes neuronales convolucionales es habitual encontrar una capa totalmente conectada, que comparte la estructura de una red neuronal tradicional con una o dos capas

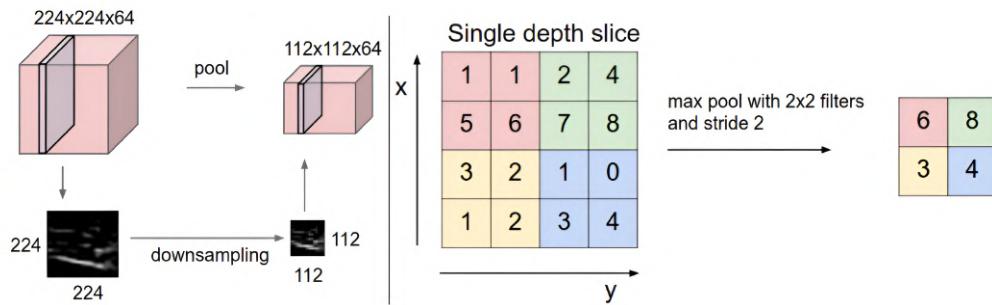


Figura 6.9.: La capa de pooling reduce espacialmente el tamaño del volumen de entrada de forma independiente en cada capa de profundidad. Izquierdo: En este ejemplo, el volumen de entrada de tamaño $224 \times 224 \times 64$ se reduce mediante un filtro de tamaño 2 y un stride de 2, obteniendo un volumen de salida de tamaño $112 \times 112 \times 64$. Nótese que la profundidad del volumen se mantiene. Derecha: La operación de submuestreo más común es el max-pooling, que en este caso se realiza con un stride de 2. Es decir, se toma el valor máximo de entre 4 números (un pequeño cuadrado de 2×2). Imagen extraída de [[Unizo](#)].

ocultas, generalmente, y está diseñada para recibir un vector de entrada con una dimensión específica (véase Figura 6.10).

Como se ha comentado en secciones anteriores, una red neuronal convolucional procesa las dimensiones del volumen de entrada de manera independiente, salvo en las capas totalmente conectadas. Estas capas totalmente conectadas suelen ser las que determinan el volumen de entrada que necesita la red para que funcione de forma adecuada.

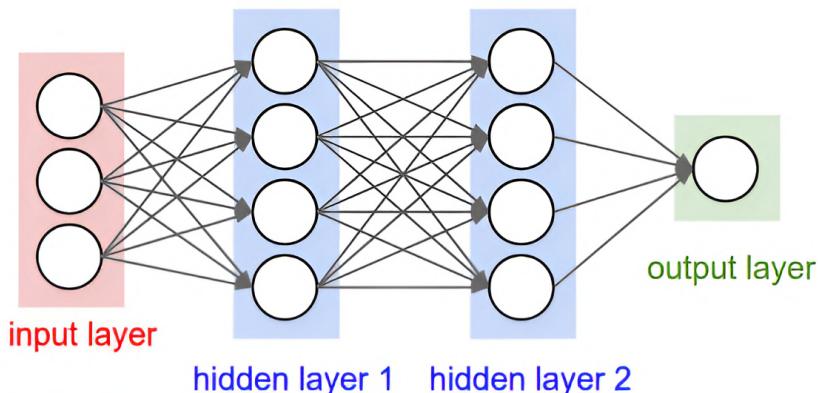


Figura 6.10.: Ejemplo de capa totalmente conectada con dos capas ocultas. Imagen extraída de [[Unizo](#)].

6. Fundamentos Teóricos

6.3.4. Batch normalization

Hoy en día, en las redes neuronales convolucionales es bastante común implementar batch normalization [IS15], una técnica que permite normalizar los datos de entrada en cada capa convolucional. Su propósito es lograr que las operaciones de convolución sean independientes entre sí. Esto implica que la distribución de los datos de entrada a cada capa no dependa de los parámetros aprendidos en la capa previa. Además, esta técnica contribuye a reducir el riesgo de sobreajuste en la red, favoreciendo la regularización.

6.3.5. Proceso de entrenamiento

Con los conceptos anteriores en mente, estamos listos para entender el proceso tradicional de entrenamiento de una red neuronal convolucional.

Normalmente, el entrenamiento se realiza usando conjuntos de imágenes llamados *mini-batches*, los cuales:

- Primero, pasan por un proceso de propagación hacia adelante a través de toda la red, en el cual se calculan las activaciones y los errores de salida, en lo que se denomina *forward pass*.
- A continuación, se calculan los gradientes de cada unidad desde la salida hasta la entrada, lo cual se conoce como *backward pass*, utilizando el algoritmo de back-propagation.
- Finalmente, los pesos se actualizan según el gradiente calculado en el paso anterior y de acuerdo con el optimizador que se esté utilizando (por ejemplo, gradiente descendente estocástico, Adam, etc.).

6.3.6. Fine-tuning

El fine-tuning es una técnica ampliamente utilizada en el Aprendizaje Profundo, especialmente en redes neuronales convolucionales, para adaptar un modelo previamente entrenado a una nueva tarea o conjunto de datos [PY10]. Esta técnica es útil cuando se dispone de un modelo entrenado en una gran base de datos que es similar a la del nuevo problema que se desea abordar, pero con diferencias específicas que requieren cierta personalización. En lugar de entrenar un modelo desde cero, el fine-tuning permite ajustar algunos parámetros del modelo para que se adapte a los nuevos datos con mayor precisión y eficiencia [YCLB14].

El proceso de fine-tuning generalmente se realiza descongelando una o más capas superiores del modelo preentrenado, mientras que las capas inferiores permanecen fijas [Ben12]. Las capas superiores se entranan entonces en el nuevo conjunto de datos, ajustando así los pesos de estas capas para reflejar las características propias de la nueva tarea. Las capas inferiores suelen retener las características generales aprendidas en el entrenamiento inicial, como bordes, texturas o patrones básicos, lo que facilita la transferencia de conocimiento a la

nueva tarea. Este enfoque reduce considerablemente el tiempo de entrenamiento y el número de datos necesarios, ya que el modelo conserva conocimientos previos que no es necesario recalcular desde cero.

En el contexto de Visión por Computador, el fine-tuning se emplea comúnmente para adaptar modelos entrenados en bases de datos generales como ImageNet a tareas específicas, como la detección de objetos en un entorno particular o la clasificación de imágenes médicas [HAE16]. Este método ha demostrado ser efectivo en numerosos estudios, aumentando la precisión y reduciendo los requerimientos computacionales en comparación con el entrenamiento desde cero [KSL19].

6.4. Visión por Computador

La Visión por Computador (*Computer Vision* en inglés) es una disciplina que combina varias áreas del conocimiento, como la Inteligencia Artificial y el Aprendizaje Automático, con el objetivo común de procesar imágenes utilizando un ordenador. El propósito es que la máquina sea capaz de extraer información relevante de las imágenes, de manera similar a como lo haría un ser humano [Ros88]. Los problemas clásicos de esta disciplina incluyen la detección de objetos o personas en las imágenes, la segmentación y la clasificación (véase Figura 6.11).

En los últimos años, el campo de la Visión por Computador ha ganado un notable impulso en la comunidad científica, en gran parte debido al avance del Aprendizaje Profundo y su herramienta principal: las redes neuronales convolucionales, que han permitido el desarrollo de aplicaciones altamente eficientes en el procesamiento de imágenes.

6.4.1. Clasificación de imágenes

El objetivo de esta primera tarea en Visión por Computador es identificar las diferentes categorías presentes en una imagen (véase Figura 6.12). Tradicionalmente, esta tarea se ha abordado utilizando una amplia gama de algoritmos de aprendizaje automático, tales como máquinas de vectores de soporte o k -nearest neighbours [Kan15].

Con la aparición del Aprendizaje Profundo y, en particular, de las redes neuronales convolucionales, el rendimiento en esta tarea mejoró notablemente en comparación con los modelos tradicionales [Oua17]. El primer modelo de aprendizaje profundo para la clasificación de imágenes que captó la atención de la comunidad internacional fue AlexNet, una red neuronal convolucional que utiliza cinco filtros convolucionales consecutivos, capas de max-pooling y tres capas totalmente conectadas al final. Esta red, propuesta por A. Krizhevsky en [KSH17], logró reducir la tasa de error de algoritmos previos del 26.2 % al 15.3 % en el desafío de ImageNet de 2012 [Oua17].

Considerando el año 2012 como un hito para la tarea de clasificación de imágenes y, en general, para la Visión por Computador, cada año se publican diversos modelos con el objetivo de superar a sus predecesores. En 2015, K. Simonyan y A. Zisserman propusieron su modelo

6. Fundamentos Teóricos

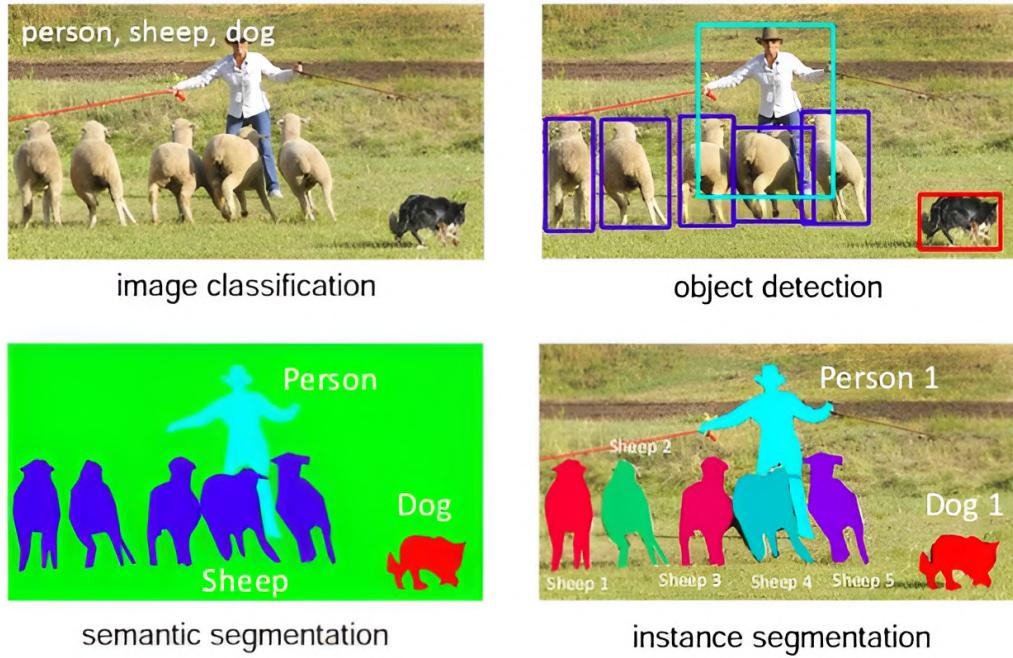


Figura 6.11.: Diferentes tareas de Visión por Computador: Clasificación de imágenes, detección de objetos, segmentación semántica y segmentación de instancias aplicadas en una misma escena. Imagen extraída de [LOW⁺19].

VGG16 [SZ14], compuesto por 16 capas convolucionales, varias capas de max-pooling y, finalmente, tres capas totalmente conectadas. En este modelo, las capas convolucionales están encadenadas con activaciones ReLU, lo que permite realizar transformaciones no-lineales y, por ende, detectar patrones más complejos. Además, redujeron el tamaño del filtro en las capas convolucionales de los 11×11 usados en AlexNet a 3×3 , demostrando que era posible reconocer los mismos patrones mientras se reducía la cantidad de parámetros a entrenar. Con esta nueva arquitectura, la tasa de error se redujo al 7,3 %.

Por otro lado, M. Lin et al. [LCY13] propusieron el concepto de módulo inception. Las capas convolucionales tradicionales emplean transformaciones lineales con funciones de activación no-lineales. Sin embargo, al utilizar un módulo inception, las transformaciones también se vuelven no-lineales. Un módulo inception consiste en entrenar varias capas convolucionales al mismo tiempo y apilar los mapas de características resultantes, vinculados a un perceptrón multicapa. C. Szegedy et al. [SLJ⁺14] utilizaron esta idea en el desarrollo de la red GoogLeNet o Inception V1.



Figura 6.12.: Resultado de aplicar clasificación de imágenes en una fotografía que contiene dos categorías: Caballo y Persona. Imagen extraída de [Sor18].

6.4.2. Detección de objetos

Esta segunda tarea de Visión por Computador se enfoca en desarrollar modelos que, al recibir una imagen como entrada, son capaces no solo de localizar espacialmente objetos de diversas clases predefinidas, sino también de asignar la clase correcta a cada objeto identificado en la imagen (véase Figura 6.13). Para ubicar los distintos objetos en una imagen, se utilizan las coordenadas del rectángulo más pequeño que los encierra, conocido como *bounding box* o caja delimitadora, que sirve como un límite para indicar dónde se encuentra el objeto en el espacio.

Durante la última década, las redes neuronales convolucionales han ganado gran popularidad en el campo de la detección de objetos [GDDM13]. Sin embargo, emplear una red neuronal convolucional clásica con una capa totalmente conectada al final para detectar múltiples objetos en una imagen no resulta viable debido a dos principales dificultades. Primero, el número de objetos a detectar en una imagen es desconocido de antemano, lo que provoca una variabilidad en la longitud de la capa de salida. Segundo, incluso si se aplicara una red neuronal convolucional a cada región de interés de la imagen para determinar si contiene un objeto, los objetos suelen variar en tamaño y ubicaciones espaciales dentro de la imagen. Esto obliga a analizar una enorme cantidad de regiones de interés, lo que resulta computacionalmente impracticable [Roh18].

No obstante, dos grupos distintos de métodos han logrado superar estos inconvenientes de formas diferentes: los métodos de una etapa y los métodos de dos etapas. Los métodos de una etapa son aquellos que emplean una red neuronal convolucional de propagación hacia adelante para determinar la ubicación de los objetos de interés, es decir, las coordenadas de las bounding boxes. Entre estos métodos encontramos modelos como YOLO [RDGF15], Multibox [ESTA13], AttentionNet [YPL⁺15], G-CNN [NRD15] o SSD [LAE⁺15]. Estos modelos no necesitan generar propuestas de región, lo que los hace más simples y rápidos. Sin embargo, esta simplicidad también introduce problemas de rendimiento, como dificultades

6. Fundamentos Teóricos

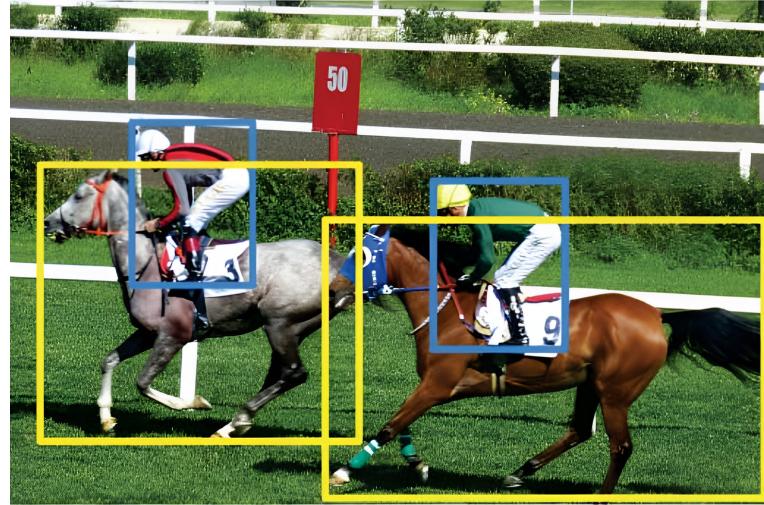


Figura 6.13.: Resultado de aplicar detección de objetos en una fotografía que contiene dos categorías: Caballo y Persona. Las bounding boxes azules delimitan a las personas detectada, mientras que las amarillas localizan a los caballos en la imagen. Imagen extraída de [Sor18].

para detectar objetos pequeños o al intentar realizar tareas más complejas como la predicción de máscaras [Jor18a].

Por otro lado, tenemos los métodos de dos etapas, como, por ejemplo, R-CNN [GDDM13], SSP-Net [HZRS14], Fast R-CNN [Gir15], FPN [LDG⁺16], Faster R-CNN [RHGS15] o R-FCN [DLHS16]. Estos modelos utilizan una red neuronal convolucional basada en regiones como primer paso. Una red neuronal convolucional basada en regiones toma como entrada una imagen y produce diferentes regiones de interés donde los objetos podrían estar ubicados. Luego, el vector de características extraído de cada región propuesta se utiliza como entrada para una serie de capas totalmente conectadas que generan una clasificación entre las distintas clases y una puntuación de confianza. La puntuación de confianza ayuda a crear un orden de prioridad entre las propuestas, de modo que solo se consideran las más confiables.

En 2014, R. Girshick et al. [GDDM13] propusieron el modelo R-CNN para resolver el problema de tener que seleccionar un número excesivo de regiones de interés. Para lograrlo, R-CNN utiliza muestreo selectivo, tal como se define en [USGS13], para generar solo 2000 propuestas de región por imagen. Luego, el tamaño de cada región se ajusta y se utiliza como entrada de una red neuronal convolucional. El resultado de esta red es un vector de características de 4096 dimensiones, que luego se introduce en una máquina de vectores de soporte para determinar la presencia o ausencia de un objeto en esa región. Este modelo logró superar a modelos anteriores en más de un 30 % cuando se aplicó al conjunto de datos PASCAL VOC 2012 [EEG⁺15], [ZZXW18].

A pesar de superar a modelos anteriores, R-CNN presenta tres inconvenientes evidentes. Primero, analizar 2000 propuestas de regiones por imagen provoca que el tiempo de entrenamiento sea bastante extenso. Segundo, el tiempo de inferencia de la red es de más de

40 segundos por imagen, lo cual impide que el modelo sea adecuado para aplicaciones en tiempo real. Finalmente, el algoritmo de muestreo selectivo era predefinido, lo que generaba una falta de aprendizaje durante esta primera etapa [ZZXW18].

Los creadores de R-CNN solucionaron algunos de estos problemas al proponer la red Fast R-CNN. Lograron hacer que la red fuera más rápida simplemente cambiando el orden de las capas. En lugar de alimentar a la red neuronal convolucional con las regiones de interés, la red recibe la imagen completa como entrada, generando un mapa de características de esta. Luego, las diferentes regiones de interés se seleccionan directamente desde el mapa de características. Estructurando la red de esta forma, la operación de convolución solo se realiza una vez por imagen, en lugar de 2000 veces [Gir15].

A pesar de las mejoras logradas por R. Girshick et al. en [Gir15], S. Ren et al. aún vieron espacio para optimizaciones, y propusieron la red Faster R-CNN [RHGS15]. Notaron que el proceso de búsqueda selectiva, utilizado tanto en Fast R-CNN como en R-CNN para generar las regiones de interés, era lento y consumía mucho tiempo, por lo que decidieron introducir su propio algoritmo de propuestas de regiones. En lugar de emplear búsqueda selectiva, propusieron el uso de una red neuronal adicional, conocida como red de propuestas de regiones (*region proposal network* en inglés), para identificar las propuestas de regiones. Más adelante profundizaremos con más detalles en las arquitecturas de R-CNN, Fast R-CNN y Faster R-CNN.

Hoy en día, los métodos de dos etapas se consideran una de las mejores opciones para la detección de objetos, ya que superan a otros enfoques en precisión [Pin17].

6.4.3. Segmentación semántica

En esta tercera tarea de Visión por Computador, el objetivo es asignar una clase a cada píxel de la imagen de entrada. En este caso, la noción de distinguir entre objetos individuales o instancias se pierde (véase Figura 6.14).

Uno de los primeros enfoques para la segmentación semántica utilizando Aprendizaje Profundo fue mediante la clasificación por parches. A cada píxel se le asignaba una clase teniendo en cuenta un parche de la imagen alrededor de él. Este método permitió el uso de capas completamente conectadas en la red empleada, ya que el tamaño del parche era fijo [Chi17].

En 2015, J. Long et al. publicaron un trabajo sobre el uso de redes completamente convolucionales para la segmentación semántica [LSD14], popularizando la idea de emplear redes convolucionales de extremo a extremo para la segmentación semántica. Se sustituyeron las capas completamente conectadas por capas completamente convolucionales, lo que permitió el uso de imágenes de distintos tamaños como entradas y obtener como salida mapas completos en lugar de clasificaciones simples. Esta arquitectura sigue una estructura de codificador-decodificador. El módulo codificador es una red de extracción de características que ya está entrenada para tareas de clasificación de imágenes, como puede ser, por ejemplo, VGGNet, ResNet o DenseNet [Jor18b]. Este codificador reduce la resolución de la imagen de entrada a través de convoluciones para obtener mapas de características de menor resolución que capturen el contexto. Luego, sigue el módulo decodificador, el cual realiza un proceso de

6. Fundamentos Teóricos



Figura 6.14.: Resultado de aplicar segmentación semántica en una fotografía que contiene dos categorías: Caballo y Persona. Todos los píxeles que pertenecen a la misma clase se muestran en una misma máscara, sin diferenciar entre instancias. Imagen extraída de [Sor18].

aumento de resolución proyectando estos mapas a escala de píxeles, produciendo un mapa de segmentación a resolución completa. El objetivo final es recuperar la información espacial [Chi17], [Jor18a]. Otra innovación relevante en este trabajo fue la inclusión de conexiones de salto. El módulo decodificador reduce la resolución de la imagen durante la fase de reducción, haciendo que las segmentaciones resultantes sean algo más toscas. Las conexiones de salto permiten transportar información desde el módulo codificador hasta el decodificador, facilitando la reconstrucción de los límites de segmentación [Jor18a].

O. Ronneberger et al. [RFB15] propusieron una nueva arquitectura mediante la extensión de las redes completamente convolucionales para mejorar la segmentación en imágenes médicas. El modelo que desarrollaron fue U-Net, que presenta dos diferencias clave respecto a su predecesor. En primer lugar, U-Net es simétrico. Durante la fase de reducción de escala, la red aprende el mapeo de características; luego, en la fase de aumento de escala, reutiliza lo aprendido previamente. Los mismos mapas de características generados en la fase de reducción se aplican durante la fase de aumento para expandir el vector de características a una imagen segmentada. El uso de los mismos mapas de características es lo que otorga simetría a la red. La segunda diferencia entre las redes completamente convolucionales y U-Net es que las conexiones de salto entre las fases de reducción y aumento de escala no aplican una suma, sino una concatenación [RFB15].

Los modelos de segmentación semántica que actualmente se consideran de vanguardia han seguido el camino establecido por las redes completamente convolucionales y U-Net. Todos presentan una estructura dividida en una etapa de extracción de características seguida de una fase de procesamiento a múltiples escalas. Una de las diferencias principales radica en el uso de bloques de mayor complejidad. En U-Net, estos bloques están formados por capas convolucionales apiladas. M. Drozdzal et al. [DVC⁺16] decidieron reemplazar estos bloques

convolucionales por bloques residuales. Los bloques residuales implementan conexiones de salto dentro del mismo bloque, no solo entre los módulos de reducción y aumento de escala. Estas conexiones contribuyen a una convergencia más rápida durante el entrenamiento. Por otro lado, S. Jegou et al. [JDV⁺16] propusieron el uso de bloques densos. Las conexiones densas permiten transportar características de bajo nivel desde las primeras capas a las más avanzadas, logrando un uso más eficiente de las características extraídas.

6.4.4. Segmentación de instancias

Esta última tarea de Visión por Computador representa la evolución natural de las dos anteriores. Mientras que la detección de objetos busca identificar las distintas entidades en una imagen y la segmentación semántica pretende asignar una clase a cada píxel generando una máscara para cada categoría, la segmentación de instancias se plantea resolver ambas tareas de forma simultánea, localizando instancias específicas de objetos con precisión a nivel de píxel (véase Figura 6.15).



Figura 6.15.: Resultado de aplicar segmentación de instancias en una fotografía que contiene dos categorías: Caballo y Persona. Todos los píxeles que pertenecen a la misma instancia se muestran en una misma máscara. El resultado final es una máscara por instancia. Imagen extraída de [Sor18].

En la segmentación de instancias, cada píxel se clasifica en una de las clases predefinidas, como en la segmentación semántica. Sin embargo, se realiza además una detección de objetos, lo que implica que cada máscara resultante no incluirá todos los objetos de una misma clase, sino solo una instancia de la clase. Esto permite que los objetos de una misma clase tengan máscaras diferentes. Esta tarea permite localizar múltiples instancias de la misma clase en una imagen.

Al igual que en las tareas previas, utilizando redes neuronales convolucionales como base común, se han propuesto diversos marcos a lo largo de los años con el objetivo de abordar

6. Fundamentos Teóricos

este problema.

B. Hariharan et al. [HAGM14b] fueron de los primeros en abordar la tarea de segmentación de instancias, a la que denominaron como detección y segmentación simultánea. Su idea fue extender la arquitectura R-CNN para obtener una máscara de cada instancia, y no solo un bounding box. Su marco de trabajo comenzó con la generación de diferentes propuestas. Utilizaron agrupación combinatoria multiescala [PAB⁺15] para proponer 2000 regiones por imagen. En un segundo paso, una red neuronal convolucional extraía características de cada una de estas regiones propuestas. Posteriormente, se entrenó una máquina de vectores de soporte para utilizar las características extraídas, obteniendo una probabilidad para cada una de las clases predefinidas. Finalmente, aplicaron supresión de no-máximos a las propuestas (consiste en seleccionar solo la propuesta con la mayor puntuación de probabilidad dentro de un área específica, mientras se descartan las demás propuestas que se solapan en gran medida, de modo que solo se conserva la propuesta más relevante para cada objeto identificado), manteniendo de esta manera solo una propuesta por instancia.

Un año después, los mismos autores mejoraron su modelo [HAGM14a]. Se dieron cuenta de que usar únicamente la salida de la última capa como representación de características causaba la pérdida de información espacial relevante. Sin embargo, las capas previas contenían información espacial más rica, aunque no capturaban la semántica. Para resolver este problema, introdujeron el concepto de hipercolumnas. En lugar de utilizar solo la salida de la última capa de la red, cada píxel se definía mediante un vector de las activaciones de todas las capas de la red neuronal convolucional por encima de ese píxel.

Otros marcos siguen dos pasos principales: uno inicial para propuestas de regiones y un segundo paso en el que se clasifican dichas propuestas. P. Pinheiro et al. [PCD15] propusieron un nuevo enfoque para generar propuestas de región. Su algoritmo, conocido como DeepMask, toma un parche de imagen como entrada y genera dos salidas diferentes. Por un lado, una máscara que no depende de la clase, y por otro lado, una puntuación que determina la probabilidad de que el parche contenga un objeto centrado y completo. El núcleo del modelo es una ConvNet que emplea una función de coste para optimizar ambas salidas simultáneamente. Este modelo logró reducir el número de propuestas generadas mientras mejoraba el rendimiento en comparación con modelos previos.

Como se mencionó anteriormente, las redes completamente convolucionales han demostrado ser muy efectivas en la tarea de segmentación semántica. Sin embargo, estas redes no proporcionan información a nivel de instancias. Para abordar este problema, J. Dai et al. [DHL⁺16] propusieron el modelo InstanceFCN, basado en las redes completamente convolucionales y capaz de segmentar distintas instancias de una misma clase. La salida tradicional de una red completamente convolucional es un mapa de puntuaciones del mismo tamaño que la imagen de entrada, donde cada valor representa la clasificación del píxel correspondiente en la imagen de entrada. Los autores de este trabajo introducen el concepto de posiciones relativas. En su modelo, cada píxel de salida actúa como un clasificador de posiciones relativas de instancias. Esto implica que la clasificación no solo se realiza entre clases, sino también utilizando posiciones relativas como lado izquierdo o fondo. De esta manera, cada píxel se clasifica teniendo en cuenta si forma parte de una posición relativa de una instancia. Estas posiciones relativas se definen mediante cuadrículas de diferentes tamaños. Debido al uso de coherencia local (predicciones consistentes para el mismo píxel cada vez que se

desplaza la ventana deslizante), no solo se reduce el coste computacional en este modelo en comparación con DeepMask, sino que también se logra una reducción significativa en el número de parámetros necesarios para el entrenamiento.

No obstante, InstanceFCN también presentó algunos inconvenientes que Y. Li et al. buscaron resolver en su estudio [LQD⁺16]. InstanceFCN no consideraba las categorías semánticas, realizaba la segmentación y detección en etapas separadas y la solución no era un modelo de extremo a extremo. La técnica de ventanas deslizantes empleada tenía un tamaño fijo y el proceso para encontrar instancias en diferentes escalas resultaba lento. Y. Li et al. propusieron la segmentación de instancias totalmente convolucional, siendo la primera solución de extremo a extremo para esta tarea. Las características extraídas y los mapas de puntuación se compartían entre las subtareas de segmentación y detección, lo que redujo el número de parámetros. Además, utilizaba propuestas de bounding boxes en lugar de ventanas deslizantes, haciendo el proceso más eficiente.

A pesar de las mejoras implementadas por Li et al. [LQD⁺16], el modelo de segmentación de instancias totalmente convolucional aún presentó algunas deficiencias. Se observaban problemas e inexactitudes cuando dos o más instancias se solapaban, generando bordes falsos incluso cuando la textura del fondo era uniforme.

Otro enfoque de investigación abordó el problema desde una perspectiva diferente. Primero realizaron segmentación semántica y luego intentaron separar las diferentes instancias para cada clase. Este es el caso de [KLA⁺16], [BU16] y [LJFU17].

En [KLA⁺16], A. Kirillov et al. propusieron InstanceCut. Su marco de trabajo tenía dos elementos distintos. Por un lado, resolvieron la tarea de segmentación semántica usando una red neuronal convolucional estándar y obteniendo una solución independiente de instancias. Por otro lado, emplearon una red neuronal convolucional distinta para obtener los bordes de las diferentes instancias. Luego, el resultado de estas dos redes neuronales convolucionales se combinaba. Entre las diferentes ventajas de este marco, se puede observar que se entrena dos modelos distintos, uno para la segmentación semántica y otro para la detección de bordes de instancia. Esto permite trabajar con características globales de las instancias de diferentes clases. Además, ayuda a que el marco sea más modular. Los avances en la segmentación semántica o en los campos de detección de bordes de instancia se pueden implementar directamente en esta solución. La principal limitación de este método es que los objetos que no están conectados no pueden considerarse como pertenecientes a la misma instancia.

Los autores de [BU16] decidieron explotar el potencial de una técnica de agrupación clásica conocida como transformación de cuencas. La idea detrás de esta transformación es que una imagen en escala de grises se puede considerar como una superficie topográfica. Luego, la superficie se inunda desde diferentes puntos localizados en sus mínimos. Si no dejamos que el agua de diferentes fuentes se mezcle, obtenemos una segmentación de los diferentes componentes de la imagen. M. Bai et al. [BU16] propusieron usar una red neuronal para aprender la energía de la transformación y así obtener segmentaciones que contengan solo una instancia. De esta manera, las diferentes instancias pueden segmentarse simplemente cortando niveles de energía únicos. Como en [KLA⁺16], una de las limitaciones principales de este marco es la imposibilidad de tratar con occlusiones.

6. Fundamentos Teóricos

En [LJFU17], se propuso el uso de redes de agrupamiento secuencial para llevar a cabo la segmentación de instancias. Las redes de agrupamiento secuencial emplean una concatenación de redes neuronales, donde cada una de ellas se enfoca en resolver subproblemas de distinta complejidad semántica dentro de la tarea de segmentación de instancias. La segmentación se divide en varias tareas más simples, y cada red neuronal se encarga de una de ellas. El propósito de la primera red es agrupar píxeles a lo largo de las filas y columnas de la imagen, prediciendo los puntos de interrupción de los objetos. Estos grupos forman segmentos de línea, los cuales sirven como entrada para la segunda red. El objetivo de esta segunda red es reunir los diferentes segmentos de línea en elementos conectados. Luego, una última red neuronal une los diversos componentes generando máscaras de instancias de los objetos. Dos de las principales limitaciones de este enfoque son el bajo rendimiento al segmentar instancias pequeñas y la segmentación ocasional de varias instancias juntas cuando estas se superponen.

Mask R-CNN [HGDG17] se propuso como una extensión de Faster R-CNN. Los autores del artículo plantearon añadir una rama suplementaria a la arquitectura de Faster R-CNN. La rama original estaba destinada a la clasificación y a la regresión de bounding boxes, obteniendo como salida las bounding boxes y las etiquetas de clase de los objetos que contienen. Por otro lado, la nueva rama adicional tenía como objetivo predecir máscaras de segmentación en cada región de interés, produciendo como resultado la máscara de la instancia contenida en cada bounding box. Mask R-CNN logró superar a todos los modelos anteriores de última generación utilizados para la tarea de segmentación de instancias en el conjunto de datos COCO [LMB⁺14]. Más adelante, proporcionaremos información más detallada sobre la arquitectura de este modelo.

6.5. Arquitecturas R-CNN

En esta sección vamos a describir en detalle las arquitecturas de Redes Neuronales Convolucionales Basadas en Regiones (R-CNN, por sus siglas en inglés), una de las primeras aproximaciones exitosas en la detección de objetos mediante Aprendizaje Profundo. Estas arquitecturas combinan redes neuronales convolucionales con métodos de propuestas de región, permitiendo localizar y clasificar objetos dentro de una imagen con mayor precisión. Se explorarán las variantes principales de esta familia de modelos, incluyendo R-CNN, Fast R-CNN y Faster R-CNN.

6.5.1. Arquitectura R-CNN

R-CNN [GDDM13] fue uno de los primeros modelos de detección de objetos basados en Aprendizaje Profundo. La combinación de una red neuronal convolucional con un algoritmo de búsqueda selectiva permitió que este modelo obtuviera buenos resultados en el conjunto de datos PASCAL VOC 2007 [EFG⁺15]. Mediante un agrupamiento de abajo hacia arriba, el algoritmo de búsqueda selectiva utilizado por R-CNN segmenta la imagen de entrada, calcula la similitud en función de características como color, tamaño y textura, fusiona regiones de alta similitud y genera las regiones finales propuestas tras una iteración continua.

La estructura de red de R-CNN se ilustra en la [Figura 6.16](#). En primer lugar, se generan un gran número de regiones propuestas independientes en la imagen de entrada (unas 2000 aproximadamente) usando el algoritmo de búsqueda selectiva. Estas propuestas, llamadas Regiones de Interés (RoI, por sus siglas en inglés), que pueden contener objetos, se ajustan a un tamaño uniforme mediante recorte o deformación y se introducen en una red AlexNet para extraer sus características. A partir de ahí, se emplean múltiples máquinas de vectores de soporte para completar la clasificación, mientras que la bounding box se ajusta combinando esta técnica con una regresión lineal.

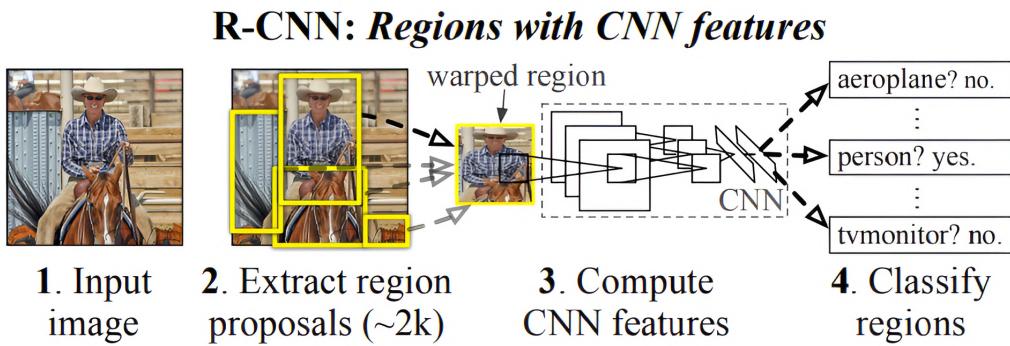


Figura 6.16.: Arquitectura R-CNN. Imagen extraída de [GDDM13].

Aunque esta arquitectura presenta una gran mejora en precisión en comparación con las arquitecturas tradicionales de detección de objetos, aún presenta varias limitaciones. Por un lado, R-CNN ajusta el tamaño de la región propuesta a una escala fija, lo cual deforma la región original y provoca pérdida de información en las características. Por otro lado, R-CNN extrae las características de cada región propuesta de manera individual, lo que implica un alto costo computacional y resulta en una velocidad de detección de objetos bastante lenta. Además, R-CNN no es una red de extremo a extremo, lo que hace que su entrenamiento consuma mucho tiempo y requiera un gran espacio de almacenamiento, dificultando su aplicación, por ejemplo, en el ámbito industrial.

6.5.2. Arquitectura Fast R-CNN

R. B. Girshick, uno de los autores de R-CNN, continuó mejorando este modelo con nuevas incorporaciones. Basándose en la estructura de red de SPP-Net (*Spatial Pyramid Pooling Network* en inglés), diseñó el algoritmo Fast R-CNN [[Gir15](#)]. La estructura de Fast R-CNN se muestra en la [Figura 6.17](#), y su principal innovación radica en la incorporación de una capa de Pooling de Regiones de Interés (RoI Pooling, por sus siglas en inglés) en la capa totalmente conectada y en la capa convolucional anterior, para extraer las características sugeridas de cada región. Los mapas de características de diferentes tamaños que ingresan a esta capa se normalizan a un tamaño fijo, eliminando la necesidad de recortar la imagen original para ajustarse a los

6. Fundamentos Teóricos

requisitos de la capa totalmente conectada. De esta forma, es posible preservar la información espacial de las muestras candidatas, reducir el uso de espacio en disco y mejorar la velocidad de entrenamiento. Además, Fast R-CNN introduce una función de pérdida multitarea para entrenar las tareas de clasificación y regresión en paralelo, acelerando la convergencia del modelo.

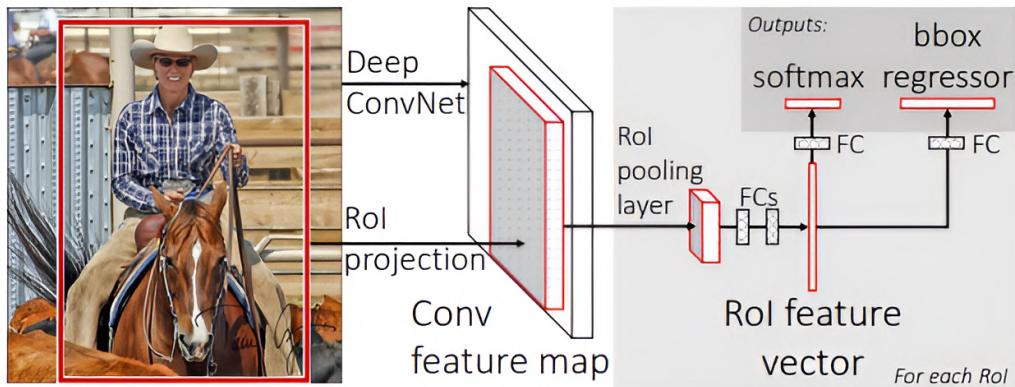


Figura 6.17.: Arquitectura Fast R-CNN. Imagen extraída de [Gir15].

Los parámetros calculados por el clasificador no necesitan almacenarse por separado, lo que ahorra espacio y simplifica la implementación del modelo.

El flujo de funcionamiento de la arquitectura Fast R-CNN es el siguiente:

1. Las regiones propuestas se generan utilizando el algoritmo de búsqueda selectiva para la imagen de entrada, y la red neuronal convolucional se emplea para extraer características de la imagen.
2. Luego, la RoI en el nuevo mapa de características obtenido se inserta en la capa de RoI Pooling para formar un vector de características unificado.
3. Finalmente, se utiliza una clasificación softmax como capa de salida para la clasificación multiclase. Además, se realiza a través de otra capa totalmente conectada, la regresión de las coordenadas de las bounding boxes.

Aunque el algoritmo Fast R-CNN ha mejorado considerablemente en precisión y velocidad en comparación con R-CNN, aún presenta algunas limitaciones. Una de ellas es la necesidad de usar un método de búsqueda selectiva que implica un elevado costo computacional para obtener las regiones propuestas. Además, el proceso de generación de las regiones propuestas sigue siendo complejo e ineficiente, ocupando gran parte del tiempo de entrenamiento y

predicción, lo cual es el principal obstáculo para mejorar la velocidad de Fast R-CNN.

6.5.3. Arquitectura Faster R-CNN

Tras una investigación exhaustiva, S. Ren et al. [RHGS15] propusieron el modelo Faster R-CNN para resolver el problema de los grandes recursos computacionales necesarios en Fast R-CNN. Para mejorar la eficiencia en la detección de objetos, este modelo incorpora una Red de Propuestas de Regiones (RPN, por sus siglas en inglés) que reemplaza el método de búsqueda selectiva. La RPN es una estructura completamente basada en redes neuronales convolucionales, cuya función principal es generar propuestas de regiones de alta calidad mediante un entrenamiento de extremo a extremo.

En Faster R-CNN, el mapa de características de la imagen de entrada se inserta en la RPN y en la rama de detección simultáneamente, lo que permite ahorrar una cantidad considerable de recursos computacionales. Como se muestra en la Figura 6.18, la estructura de Faster R-CNN se compone principalmente de tres módulos: extracción de características, RPN y regresión junto con clasificación.

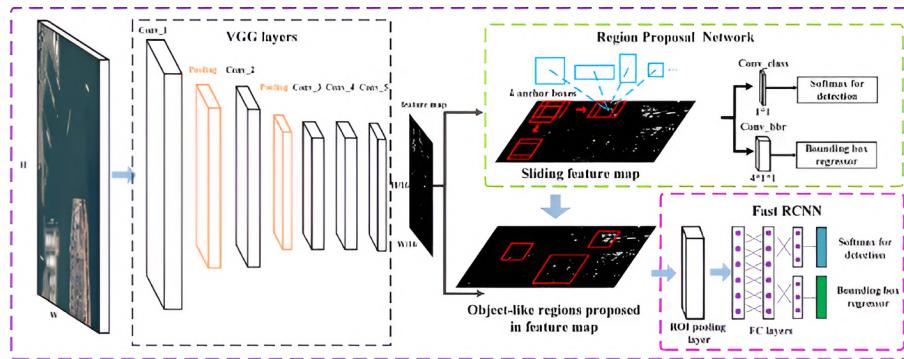


Figura 6.18.: Arquitectura Faster R-CNN. Imagen extraída de [DSZ⁺18].

El flujo de funcionamiento de la arquitectura Faster R-CNN es el siguiente:

1. La red neuronal convolucional se utiliza para obtener el mapa de características correspondiente a la imagen de entrada.
2. Luego, la RPN se encarga de clasificar tentativamente el fondo y el primer plano, generando así las regiones propuestas.
3. Al igual que en Fast R-CNN, se emplea la capa de ROI Pooling para producir un mapa de características de tamaño fijo.

6. Fundamentos Teóricos

4. Finalmente, en la rama de clasificación se obtiene la puntuación de confianza del objeto y, en la rama de regresión, se determinan las coordenadas de la bounding box del objeto.

6.6. Inteligencia Artificial Explicable

La Inteligencia Artificial Explicable (XAI, por sus siglas en inglés) es una disciplina emergente dentro de la Inteligencia Artificial que se centra en hacer que los modelos sean comprensibles y confiables para los seres humanos [AB18]. A medida que los modelos de Aprendizaje Profundo, como las redes neuronales, se han vuelto más complejos, especialmente en aplicaciones críticas como la medicina, finanzas y sistemas autónomos, la necesidad de comprender y explicar sus decisiones se ha vuelto indispensable (véase Figura 6.19). La XAI permite analizar cómo los modelos llegan a sus predicciones, lo cual es clave para generar confianza en estos sistemas y garantizar su uso ético y responsable.

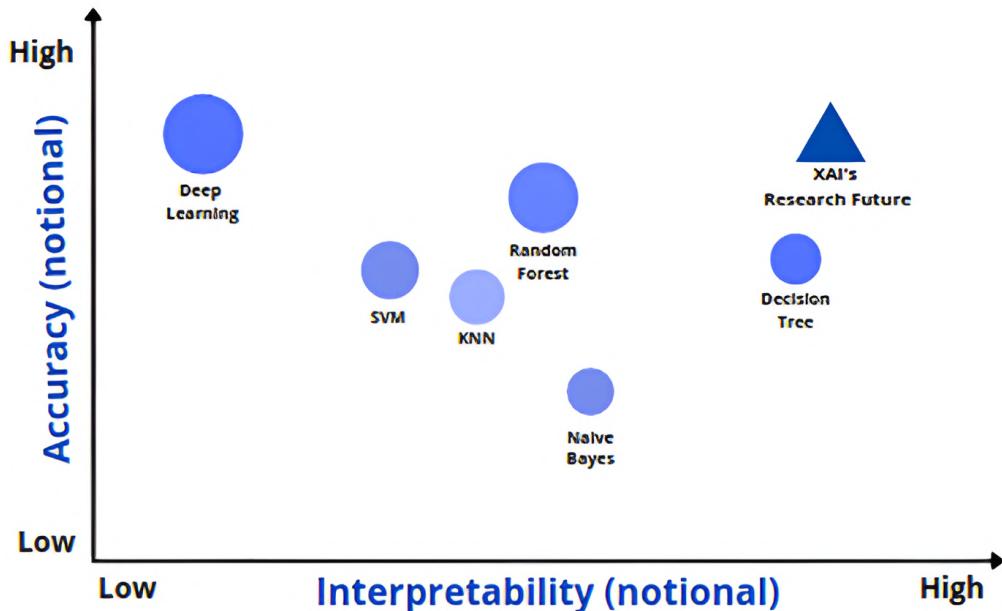


Figura 6.19.: Precisión frente a interpretabilidad para distintos algoritmos de Aprendizaje Automático. Imagen extraída de [ASJ⁺21].

6.6.1. Motivación y conceptos fundamentales

La motivación detrás de la XAI radica en la necesidad de transparencia en los sistemas de inteligencia artificial, particularmente en aquellos conocidos como modelos de caja negra debido a su complejidad y opacidad. Este tipo de modelos, como las redes neuronales

convolucionales, son eficaces para tareas complejas pero difíciles de interpretar. En este contexto, surge la necesidad de métodos que hagan visibles los razonamientos internos de estos modelos. La XAI busca no solo maximizar la precisión del modelo, sino también ofrecer interpretaciones claras de sus decisiones, lo cual es fundamental para aplicaciones en las que se exige una justificación de las decisiones de la IA, como ocurre en el ámbito médico o en la justicia.

Existen dos conceptos clave dentro de la XAI: la explicación y la interpretación. Una explicación se refiere a la capacidad del sistema para comunicar el conjunto de características específicas que influyeron en una predicción. La interpretación, por otro lado, se centra en traducir la salida del modelo a un dominio comprensible para los humanos. Esto implica simplificar la representación de las predicciones y exponer los factores que influyeron en las decisiones del modelo.

Además, en XAI se suele hacer referencia a la transparencia, que se entiende como la capacidad inherente del modelo para ser comprensible. Los modelos de caja blanca son aquellos que son transparentes y, por lo tanto, más fáciles de interpretar, mientras que los modelos de caja negra requieren de técnicas post-hoc para hacer explícitos sus razonamientos [DVK17].

6.6.2. Tipos de explicabilidad

Dentro de la XAI, existen diferentes enfoques y técnicas, los cuales pueden clasificarse en función de su aplicabilidad y su alcance. Estos métodos pueden ser categorizados de la siguiente manera en base a dos criterios [AB18]:

- **Criterio temporal:** Clasifica los métodos de explicabilidad en función del momento en que se aplica la interpretación en relación con el entrenamiento y funcionamiento del modelo.
 - **Explicabilidad intrínseca:** En este enfoque, la interpretabilidad es una propiedad inherente al modelo. Los modelos intrínsecamente explicables, como las regresiones lineales o los árboles de decisión poco profundos, son diseñados de tal forma que sus decisiones son comprensibles sin necesidad de métodos adicionales. La desventaja de estos modelos es que suelen sacrificar precisión en tareas complejas, como las abordadas por las redes neuronales profundas.
 - **Explicabilidad post-hoc:** Este enfoque se aplica a modelos complejos después de que han sido entrenados. Las técnicas post-hoc no alteran la estructura del modelo, sino que buscan entender su funcionamiento a través de análisis externos. Esta categoría incluye técnicas como LIME, SHAP y Grad-CAM (sobre la cual se profundizará en la sección dedicada a los métodos propuestos), entre otras.
- **Criterio de alcance de la explicabilidad:** Clasifica las técnicas en función del nivel de detalle o el alcance de las explicaciones que proporcionan, es decir, si se dirigen a

6. Fundamentos Teóricos

entender el modelo en su totalidad o solo se enfocan en una predicción específica.

- **Explicabilidad global:** Las explicaciones globales se centran en ofrecer una visión general del comportamiento del modelo en su totalidad, identificando las reglas o características generales que afectan sus decisiones. Este enfoque es útil para comprender el modelo desde una perspectiva amplia y suele aplicarse en modelos más simples o mediante técnicas de interpretación que resumen la lógica del modelo.
- **Explicabilidad local:** Las explicaciones locales, en cambio, se enfocan en justificar la predicción para una entrada específica. Estas técnicas ayudan a entender qué características de una entrada concreta influyeron en la decisión del modelo. Los métodos locales son particularmente útiles en modelos de caja negra, donde es difícil obtener una visión global coherente.

6.6.3. Propiedades deseables

Para que un método de explicabilidad sea efectivo, debe cumplir con ciertas propiedades [MSM18]:

- **Comprendibilidad:** Las explicaciones deben ser claras y fáciles de entender por usuarios sin conocimientos técnicos profundos en IA.
- **Fidelidad:** La explicación debe representar de manera precisa el funcionamiento del modelo, evitando distorsiones o simplificaciones que puedan inducir a error.
- **Estabilidad:** Los resultados de una técnica explicativa deben ser consistentes para una misma entrada. Es decir, las predicciones y las explicaciones deben ser reproducibles.
- **Justicia:** Un modelo explicable debe permitir identificar y corregir posibles sesgos o disparidades en sus predicciones.

6.6.4. Técnicas de explicabilidad post-hoc en redes neuronales convolucionales

Las redes neuronales convolucionales, como ya hemos visto, son modelos complejos que operan de manera efectiva en tareas de Visión por Computador, pero su falta de interpretabilidad plantea desafíos, especialmente en aplicaciones críticas. En este contexto, la XAI juega un papel fundamental, permitiendo a los expertos entender qué características o regiones de las imágenes influyen en las decisiones del modelo. Esto se logra mediante técnicas de visualización como los mapas de calor y el análisis de activación, que identifican las áreas más relevantes de una imagen.

Dada la amplitud del campo de la XAI, vamos a centrarnos en explorar las técnicas de explicabilidad post-hoc en redes neuronales convolucionales. Este enfoque supone un modelo ya entrenado, y se busca atribuir la predicción de cada muestra de datos a las características de entrada de manera significativa.

6.6.4.1. Explicaciones basadas en eliminación

Los métodos de explicación mediante eliminación se basan en la idea de evaluar la relevancia de una característica en un modelo de Aprendizaje Automático al modificar o eliminar dicha característica y analizar el impacto en las predicciones del modelo. Para ilustrar este concepto, consideremos un modelo de aprendizaje que utiliza un conjunto definido de características para realizar sus predicciones. Si al ajustar el valor de una de estas características no se observan cambios significativos en la salida del modelo, es razonable concluir que esta característica tiene poca influencia en la predicción obtenida por el modelo. Es importante resaltar que el modo en que se interpreta y gestiona la modificación o eliminación de características puede variar según el método y contexto de aplicación. A continuación, se describen algunos métodos destacados en esta categoría:

- **LIME (*Local Interpretable Model-agnostic Explanations*):** Este método permite explicar las predicciones individuales de cualquier modelo. Genera un conjunto de datos sintético alrededor de la instancia de interés y ajusta un modelo lineal a estas predicciones para proporcionar una explicación local [RSG16].
- **SHAP (*SHapley Additive exPlanations*):** Basado en la teoría de juegos, este método distribuye la contribución de cada característica a una predicción particular utilizando los valores de Shapley, que consideran todas las combinaciones posibles de características [L17].

6.6.4.2. Explicaciones basadas en gradientes

Los métodos de explicación basados en gradientes se fundamentan principalmente en el análisis del gradiente para comprender las decisiones del modelo. A diferencia de otras técnicas que requieren múltiples evaluaciones de la función mediante perturbaciones, estos métodos examinan cómo pequeñas variaciones en las entradas afectan a la salida del modelo, sin necesidad de una perturbación explícita. Empleando el algoritmo de back-propagation (esencial para el entrenamiento de redes neuronales, como ya vimos), es posible estudiar estos gradientes, obteniendo métricas y visualizaciones que destacan las características más relevantes para las predicciones del modelo. Esta metodología no solo es computacionalmente más eficiente, sino que también proporciona una visión más detallada de las decisiones del modelo desde una perspectiva global. Dentro de esta categoría, destacamos los siguientes métodos:

- **Mapas de Saliencia (*Saliency Maps*):** Calcula el gradiente de la salida del modelo respecto a la entrada, mostrando las áreas de mayor relevancia para la decisión del modelo. Es un método sencillo pero poderoso para identificar qué segmentos de una

6. Fundamentos Teóricos

imagen son cruciales para una clasificación específica [SVZ13].

- **Retropropagación Guiada (*Guided Backpropagation*):** Consiste en una variante del método de backpropagation, en el cual se ajusta el cálculo del gradiente para obtener visualizaciones más claras y coherentes de las características importantes, mejorando la legibilidad respecto a los mapas de saliencia convencionales [SDBR14].
- **Grad-CAM (*Gradient-weighted Class Activation Mapping*):** Evalúa los gradientes en la salida para una clase específica y los proyecta sobre los mapas de características de una capa intermedia (como una capa convolucional), generando un mapa de calor que destaca las regiones de la imagen más relevantes para esa clase. Este método es aplicable a cualquier arquitectura CNN sin necesidad de reentrenamiento y es uno de los métodos de XAI más utilizados en redes neuronales convolucionales. Más adelante de este trabajo, exploramos este método con más detalle [SCD⁺17].
- **Gradientes Integrados (*Integrated Gradients*):** Calcula la integral de los gradientes a lo largo de un trayecto desde un punto de referencia (como una imagen completamente blanca) hasta la entrada actual. Esto permite asignar una relevancia a cada característica de entrada en la predicción final del modelo [STY17].

6.6.4.3. Explicaciones basadas en propagación

Finalmente, esta categoría incluye métodos que, en lugar de depender solo de la salida del modelo, exploran los procesos internos de la red mediante la adaptación o reinterpretación de algoritmos clásicos, como el backpropagation, con el propósito de entender mejor cómo la red realiza sus decisiones. A continuación, describimos algunos métodos destacados en esta categoría:

- **LRP (*Layer-wise Relevance Propagation*):** Este método asigna un valor de relevancia a cada entrada, proporcionando una descomposición de la salida en función de sus entradas. La relevancia se distribuye hacia atrás a través de la red, ofreciendo una perspectiva de cómo cada neurona contribuye al resultado final [BBM⁺15].
- **DTD (*Deep Taylor Decomposition*):** Basado en la expansión de Taylor, este método busca aproximar la función de activación de la red neuronal a través de series polinómicas. Al descomponer estas funciones, este método ofrece una interpretación de la contribución de cada entrada al resultado final en términos de relevancia [MLB⁺17].

7. Estado del Arte

En esta sección, se presenta una revisión de la literatura actual (además de aquella ya citada en el [Capítulo 6](#)) relacionada con estudios sobre arquitecturas avanzadas de redes neuronales para segmentación de instancias y con enfoques recientes en XAI aplicados a tareas de Visión por Computador. Para ello, utilizaremos la base de datos Scopus con el fin de realizar la búsqueda y consulta de artículos científicos relacionados.

Para realizar esta revisión, realizamos una consulta en la base de datos científica Scopus, utilizando las siguientes palabras clave específicas para obtener artículos relacionados con el tema de estudio (restringiendo a aquellos que se sean del campo de Ciencias de la Computación):

```
TITLE-ABS-KEY ( ( "instance segmentation" ) AND  
( "Explainable AI" OR "interpretability" OR "explainability" ) ) AND  
( LIMIT-TO ( SUBJAREA , "COMP" ) )
```

Como se puede observar en la [Figura 7.1](#), existe una tendencia creciente en el número de publicaciones relacionadas con los temas de segmentación de instancias y XAI aplicados a Visión por Computador. Esta tendencia comienza a notarse a partir de 2019, con un incremento marcado en los años posteriores, lo que indica un interés creciente de la comunidad científica en investigar y desarrollar explicabilidad en arquitecturas neuronales profundas de segmentación de instancias. El pico de publicaciones en 2023 sugiere que el tema ha alcanzado una atención relevante en el ámbito de la investigación, posiblemente debido a los avances en modelos de redes neuronales y la demanda de sistemas de inteligencia artificial más transparentes y comprensibles.

Sin embargo, con dicha búsqueda anterior se han obtenido un total de 21 artículos, lo cual refleja que, aunque es un área de investigación en auge, aún no cuenta con un volumen extenso de bibliografía específica, lo cual sugiere que el campo sigue en desarrollo y existen oportunidades para contribuciones significativas.

A continuación, se describen brevemente los cinco artículos más relevantes seleccionados de entre los 21 encontrados en la búsqueda bibliográfica en Scopus. Estos artículos han sido escogidos por su relevancia en el campo de estudio que nos ocupa, ya que abordan temas clave relacionados con la segmentación de instancias y la explicabilidad en Visión por Computador. Los artículos están ordenados de manera cronológica, comenzando por los más antiguos, lo cual permite observar la evolución de los enfoques y metodologías en esta área de investigación. Estos son los siguientes:

7. Estado del Arte

Documents by year

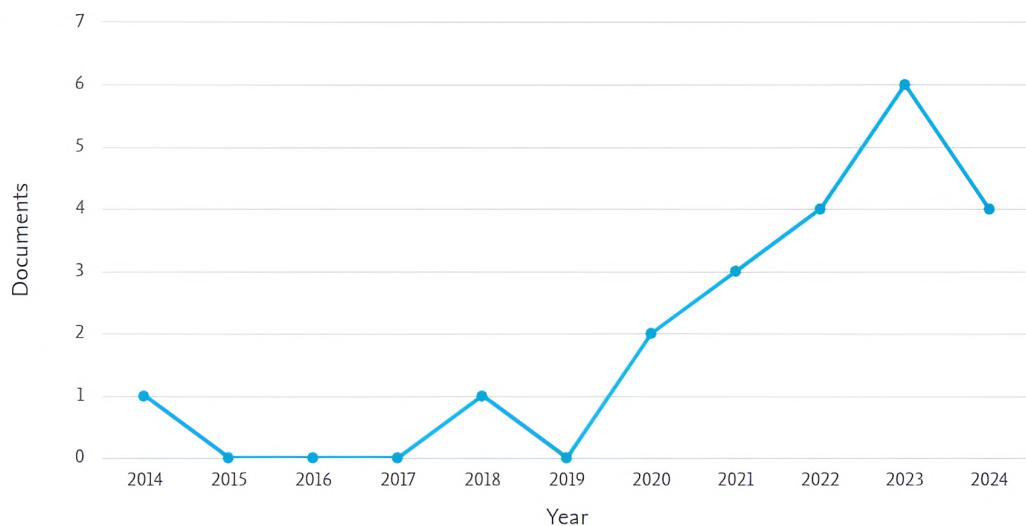


Figura 7.1.: Número de publicaciones por año obtenidas en Scopus con la búsqueda realizada el 3 de noviembre de 2024, utilizando términos de búsqueda relacionados con segmentación de instancias y XAI. Se observa un aumento significativo en las publicaciones a partir de 2019, reflejando el creciente interés en la explicabilidad aplicada a arquitecturas neuronales profundas de segmentación de instancias. Imagen extraída de [Sco24].

1. ***Explain Any Concept: Segment Anything Meets Concept-Based Explanation:*** En el artículo publicado en mayo de 2023 por A. Sun et al. [APYS23], se explora la combinación del modelo Segment Anything (SAM) con técnicas de XAI (Explainable AI) basadas en conceptos para ofrecer interpretabilidad en modelos de redes neuronales profundas en tareas de Visión por Computador. Los métodos tradicionales de XAI se enfocan en el nivel de píxeles, lo cual dificulta la comprensión, mientras que SAM permite segmentar automáticamente imágenes en conceptos reconocibles para los humanos. El enfoque propuesto, llamado Explain Any Concept (EAC), utiliza SAM para identificar conceptos en una imagen y emplea un modelo de sustitución para aproximar el modelo original, lo que reduce la carga computacional de calcular el valor de Shapley. El estudio muestra que EAC supera a métodos de interpretabilidad de píxeles y superpíxeles, logrando explicaciones más comprensibles y precisas en términos de correlación con las decisiones internas del modelo. Este trabajo contribuye a mejorar la interpretabilidad y eficiencia de XAI en redes neuronales profundas para tareas de clasificación y detección de objetos.
2. ***Unified Intersection Over Union for Explainable Artificial Intelligence:*** La investigación de J. Stodt, C. Reich, y N. Clarke [SRC24], publicada en abril de 2024, aborda la necesidad de una métrica efectiva para evaluar la precisión de los modelos de detección

de objetos que utilizan cuadros delimitadores (bounding boxes) en Visión por Computador. Los métodos tradicionales, como IoU, presentan limitaciones para diferenciar casos en que los cuadros no se superponen o están completamente contenidos uno en el otro. Los autores proponen la métrica UIoU (Unified Intersection over Union), que combina propiedades de IoU, GIoU y DIoU y agrega un factor de similitud para distinguir claramente entre tres posibles configuraciones de los cuadros: no superpuestos, superpuestos y uno dentro del otro. UIoU permite una evaluación más comprensible del rendimiento de los modelos de detección, ofreciendo una métrica más interpretativa para investigadores y usuarios de IA explicable. Esta métrica mejora la comparabilidad y precisión en la evaluación de modelos de detección de objetos, proporcionando una herramienta avanzada en el campo de la IA explicable.

3. *Improving the Explain-Any-Concept by Introducing Nonlinearity to the Trainable Surrogate Model:* En junio de 2024, M. Zaval y S. Ozer publicaron un estudio [MS24] que presenta una mejora al modelo Explain Any Concept (EAC), que se utiliza en el ámbito de la Inteligencia Artificial Explicable para interpretar decisiones de redes neuronales profundas en tareas de Visión por Computador. EAC se basa en un modelo sustituto con una capa lineal entrenable que simula la red objetivo. En esta investigación, los autores proponen introducir una capa no lineal adicional en el modelo sustituto para mejorar la precisión de las explicaciones generadas. Los experimentos muestran que la incorporación de esta no-linealidad mejora el rendimiento explicativo del modelo en los conjuntos de datos ImageNet y COCO, obteniendo mejores valores de AUC en comparación con el EAC original. Esta modificación incrementa la fidelidad de las explicaciones sin un costo significativo en la eficiencia computacional, lo que sugiere que las mejoras arquitectónicas en el modelo sustituto pueden aumentar la precisión de las explicaciones en el marco de EAC.
4. *Review of Semantic Segmentation by Using Deep Learning Methods:* Con una revisión exhaustiva de las técnicas de segmentación semántica, el trabajo B. Rajeswari et al. [RMKH24], publicado en agosto de 2024, examina el estado actual de los métodos de Aprendizaje Profundo aplicados a esta área. La segmentación semántica es una tarea fundamental en Visión por Computador que implica la clasificación de cada píxel de una imagen para asignarlo a una categoría específica. A lo largo de los años, numerosos autores han desarrollado arquitecturas innovadoras para abordar los desafíos únicos de la segmentación semántica, incluyendo modelos como SegNet y DeepLab. Los autores destacan aplicaciones de la segmentación semántica en áreas como la imagen médica, vehículos autónomos y realidad aumentada, mostrando su impacto potencial en la sociedad. Entre los temas recientes que aborda el artículo, se encuentra la integración de la segmentación semántica con otras tareas de visión, como la segmentación de instancias y la segmentación panóptica, así como la exploración de métodos para mejorar la interpretabilidad, la robustez ante cambios de dominio y la eficiencia en entornos con recursos limitados.
5. *Quantifying and Learning Static Vs. Dynamic Information in Deep Spatiotemporal Networks:* Un estudio reciente, publicado en septiembre de 2024 por M. Kowal et al. [KSI⁺24], aborda la falta de comprensión sobre la información capturada por los

7. Estado del Arte

modelos de redes neuronales profundas en tareas de vídeo, específicamente en sus representaciones intermedias. Los autores proponen una metodología para cuantificar el sesgo hacia información estática (de un solo fotograma) o dinámica (a través de varios fotogramas) en modelos espacio-temporales. Aplican su enfoque en tres tareas: reconocimiento de acciones, segmentación automática de objetos en vídeo (AVOS) y segmentación de instancias en vídeo (VIS). Entre sus principales hallazgos, los autores destacan que los modelos analizados tienden a estar sesgados hacia la información estática y que ciertos conjuntos de datos, supuestamente orientados a dinámicas, en realidad presentan sesgo hacia información estática. Además, los autores introducen StaticDropout, un mecanismo de regularización que ayuda a estos modelos a enfocarse más en la información dinámica en lugar de la estática. Este estudio contribuye al avance de la interpretabilidad en redes espacio-temporales, resaltando cómo los sesgos hacia lo estático o dinámico pueden influir en el rendimiento de las redes en diferentes tareas de Visión por Computador.

El análisis de los artículos revisados muestra que la explicabilidad en redes neuronales profundas aplicadas a segmentación de instancias ha ganado notable relevancia en los últimos años, impulsada por la creciente necesidad de sistemas de inteligencia artificial transparentes y comprensibles. A medida que los métodos de XAI y segmentación han evolucionado, se han propuesto arquitecturas y métricas más sofisticadas para mejorar la precisión y comprensión de los modelos, desde enfoques iniciales de segmentación semántica hasta métodos recientes como EAC, que aprovechan el modelo SAM para identificar conceptos explicativos de manera automática.

Por otro lado, se observa que, si bien los avances han sido significativos, el campo sigue en desarrollo, especialmente en lo que respecta a la robustez de las explicaciones ante distintos dominios y al balance entre precisión y eficiencia computacional en modelos complejos. Esto destaca la oportunidad que tiene nuestra propuesta de contribuir con un enfoque que logre explicabilidad efectiva.

8. Métodos Propuestos

En este capítulo se detallarán los métodos empleados y propuestos para llevar a cabo el desarrollo de este trabajo. La descripción de estos métodos se basa en el conocimiento adquirido a través de literatura especializada en el campo de las redes neuronales profundas y la segmentación de instancias. Además, se han utilizado diversas fuentes bibliográficas para sustentar las decisiones de diseño y metodologías de los diferentes métodos, las cuales se explicarán en los apartados correspondientes.

8.1. Mask R-CNN

Como se mencionó anteriormente, K. He et al. [HGDG17] concibieron Mask R-CNN, una arquitectura neuronal profunda para la segmentación de instancias, como una ampliación del modelo previo Faster R-CNN. La arquitectura de Mask R-CNN se muestra en la [Figura 8.1](#) y está compuesta principalmente por:

1. **Red de Extracción de Características (FEN, por sus siglas en inglés):** Red inicial encargada de procesar la imagen de entrada y generar mapas de características en múltiples escalas. Para esto, utiliza una ResNet con estructura de Red Piramidal de Características (FPN, por sus siglas en inglés), lo que permite extraer características a distintos niveles de detalle y resolver variaciones en la escala de los objetos.
2. **Red de Propuestas de Regiones (RPN, por sus siglas en inglés):** Utiliza los mapas de características generados por la FEN para identificar las regiones con mayor probabilidad de contener objetos, conocidas como regiones de interés (RoIs, por sus siglas en inglés).
3. **Capa de Alineación de Regiones de Interés (RoI Align Layer en inglés):** Ajusta todas las RoIs generadas por la RPN al mismo tamaño para facilitar su procesamiento.
4. **Ramas de predicciones y segmentación:** Se encargan de las tareas finales, como la clasificación y la regresión de las bounding boxes, así como la predicción de máscaras de segmentación para cada objeto detectado.

8.1.1. Backbone: Red de extracción de características

Para optimizar el tiempo y hacer el proceso más eficiente, Fast R-CNN incorporó el uso de una red neuronal convolucional inicial para calcular el mapa de características de la imagen

8. Métodos Propuestos

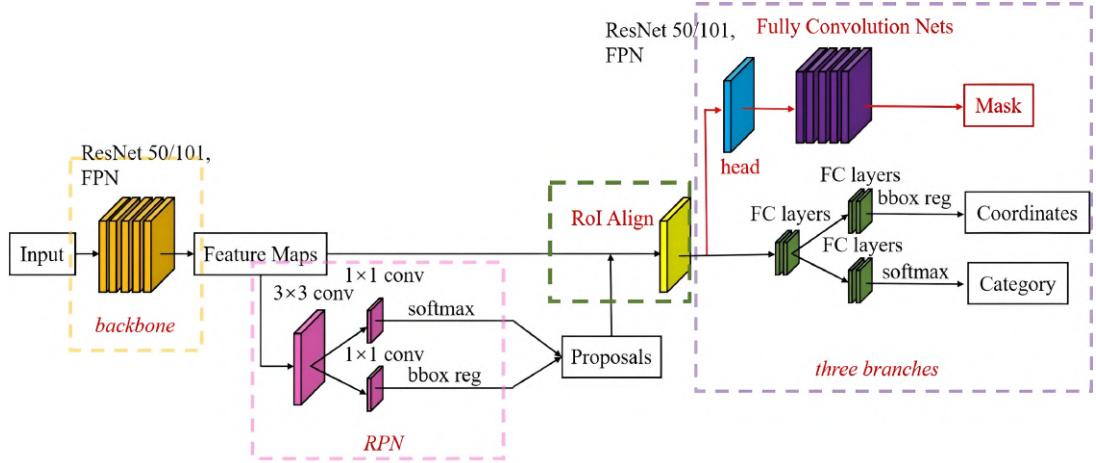


Figura 8.1.: Arquitectura Mask R-CNN. Imagen extraída de [SKY23].

completa. Este paso ahorra tiempo, ya que, de otro modo, se tendría que calcular el mapa de características de cada bounding box propuesta de forma individual. Esta red inicial se conoce como Red de Extracción de Características (FEN, por sus siglas en inglés), y es el backbone (la red principal o base que se utiliza para extraer características de la imagen de entrada) de la arquitectura Mask R-CNN.

La arquitectura convolucional del backbone se emplea como una etapa inicial para obtener mapas de características a partir de las imágenes de entrada. En el artículo original [HGDG17], los autores utilizaron una arquitectura ResNet como backbone [HZRS15] en combinación con una Red Piramidal de Características (FPN, por sus siglas en inglés) [LDG⁺16], las cuales se describen con más detalle a continuación.

8.1.1.1. ResNet

En comparación con las primeras etapas de desarrollo, las redes neuronales convolucionales han evolucionado hacia arquitecturas con una mayor profundidad, lo que permite extraer un número mayor de características y potencialmente mejorar el rendimiento del modelo. Sin embargo, en la práctica del entrenamiento, los investigadores han observado que, aunque se utilicen modelos más profundos, las mejoras en el desempeño pueden volverse menos notables o incluso disminuir. Esto se debe a que la red debe ajustarse a la complejidad específica del problema. Además, el entrenamiento de redes profundas enfrenta desafíos comunes como el sobreajuste, la desaparición del gradiente y la pérdida de capacidad de expresión no-lineal. La solución tradicional para mitigar estos problemas suele ser incrementar el número de muestras de entrenamiento, ajustar los parámetros del modelo o modificar el número de iteraciones. Sin embargo, estas soluciones suelen ser costosas en términos de tiempo y recursos, y pueden no ser efectivas.

La arquitectura ResNet [HZRS15] propone una alternativa innovadora para abordar estas

dificultades. Su principal idea de diseño se basa en la capacidad de las redes neuronales convolucionales para realizar un mapeo de identidad. La expresión funcional del mapeo de identidad se muestra en la siguiente fórmula:

$$H(x) = x.$$

Esto significa que el resultado de la salida es igual al de la entrada original. La característica principal de una red residual es que la entrada original se transmite directamente a la salida tras pasar por las capas de convolución, mediante una conexión rápida. Esto permite superar la limitación de que la salida de una capa solo puede usarse como entrada de la siguiente, como ocurría en redes neuronales convolucionales previas, como AlexNet [KSH17] o VGG [SZ14], sin necesidad de parámetros adicionales. La expresión funcional general de cada bloque residual en la red se muestra en la siguiente fórmula, donde el mapeo de identidad es una parte fundamental:

$$H(x) = F(x) + x.$$

Como se observa en la ecuación anterior, el objetivo principal de la capa de la red durante el entrenamiento es ajustar la función residual $F(x)$:

$$F(x) = H(x) + x.$$

La ventaja de este diseño es que, incluso si la red residual tiene un gran número de capas, es menos probable que experimente una degradación, siempre que no esté limitada por las condiciones de software o hardware. Además, la dificultad de aprender la función residual es menor que la de aprender la salida original, lo cual puede acelerar la convergencia del modelo.

El diseño de un bloque residual de dos capas se muestra en la [Figura 8.2](#), siendo la expresión de la función correspondiente la siguiente:

$$y = F(x, \{w_i\}) + x,$$

donde x es la entrada del bloque residual, y es la salida del bloque residual, y $F(x, \{w_i\})$ es la función residual final que debe aprenderse. La definición exacta de la función objetivo F se muestra en la siguiente fórmula:

$$F = w_2 \sigma(w_1 x),$$

donde σ representa la función de activación ReLU, mientras que w_1 y w_2 corresponden a los

8. Métodos Propuestos

pesos asociados con la capa de la red correspondiente.

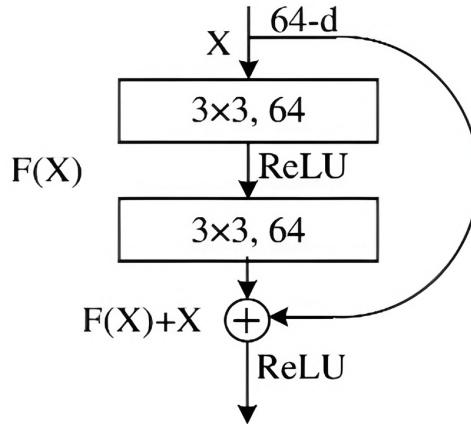


Figura 8.2.: Bloque residual de dos capas. Imagen extraída de [HZRS15].

Para simplificar la optimización de ResNet y mejorar su eficiencia, se utilizan bloques residuales de tres capas, también conocida como estructura cuello de botella o bottleneck (véase Figura 8.3). En este diseño, cada bloque residual contiene principalmente dos tipos de convoluciones: una de 1×1 y otra de 3×3 . La convolución de 1×1 se emplea primero para reducir el número de canales del mapa de características de entrada, lo que disminuye el consumo computacional innecesario al reducir la cantidad de datos procesados. A continuación, se aplica la convolución de 3×3 , que extrae características complejas de los datos. Finalmente, otra convolución de 1×1 restaura el número de canales al tamaño original, asegurando que la salida mantenga la misma profundidad que el mapa de entrada. Este diseño de tres capas, en comparación con un bloque residual de dos capas, permite reducir significativamente el número de parámetros del modelo, lo cual optimiza la memoria utilizada y acelera el entrenamiento de la red sin perder capacidad de aprendizaje.

8.1.1.2. Red piramidal de características

Como se muestra en la Figura 8.4, en Mask R-CNN, se utiliza una estructura de Red Piramidal de Características (FPN, por sus siglas en inglés) junto con una arquitectura ResNet (como ResNet101, por ejemplo, donde el número 101 hace referencia a la cantidad de capas en la arquitectura) para el backbone. La FPN permite generar mapas de características en diferentes escalas. Esta red contiene dos rutas: una ascendente y otra descendente, conectadas mediante conexiones de salto.

A medida que avanzamos en la ruta ascendente, las características extraídas tienen mayor nivel, siendo semánticamente más ricas, mientras que la resolución espacial disminuye. Las capas inferiores de esta ruta ascendente tienen una resolución mayor; sin embargo, su valor

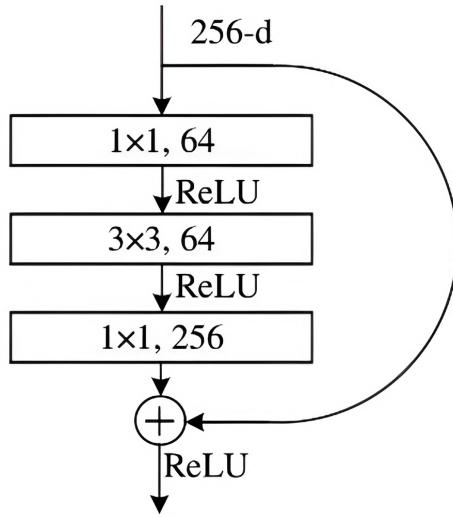


Figura 8.3.: Bloque residual de tres capas (también conocido como estructura bottleneck). Imagen extraída de [HZRS15].

semántico no es tan relevante, lo que afecta al rendimiento de la red en la detección de objetos pequeños. Para solventar este inconveniente, la FPN implementa la ruta descendente, logrando una fuerte semántica en todos los niveles de la pirámide.

El proceso de la ruta descendente se realiza mediante una interpolación de los mapas de características provenientes de niveles superiores, empleando un muestreo ascendente por vecinos más cercanos. Este método mejora los resultados al aprovechar la información que las conexiones de salto aportan desde la ruta ascendente.

El propósito de la interpolación por vecino más cercano es aumentar el tamaño de la imagen. Basado en los píxeles de la imagen original, se utiliza un algoritmo de interpolación adecuado para insertar nuevos píxeles entre los existentes. Este método es uno de los más sencillos y no requiere cálculos complejos. De los cuatro píxeles adyacentes al píxel que se quiere calcular, se asigna el valor del píxel más cercano. Si sus coordenadas son $(i + u, j + v)$, donde i, j son enteros y u, v son decimales entre 0 y 1, entonces, teniendo como referencia la Figura 8.5, $(i + u, j + v)$ cae en el área A, es decir, $(u < 0.5, v < 0.5)$, se asigna el valor del punto $a(i, j)$ al píxel que se está calculando. De manera similar, si $(i + u, j + v)$ cae en el área B, es decir, $(u > 0.5, v < 0.5)$, se asigna el valor del punto $b(i + 1, j)$ al píxel que se está calculando. Un razonamiento análogo se realiza para el caso en el que el píxel que se está calculando $(i + u, j + v)$ caiga en área C o D.

8.1.2. Red de propuestas de regiones

Como hemos visto, para facilitar la detección de objetos de diferentes tamaños, la FPN genera mapas de características en múltiples escalas a partir de la imagen de entrada, proporcionan-

8. Métodos Propuestos

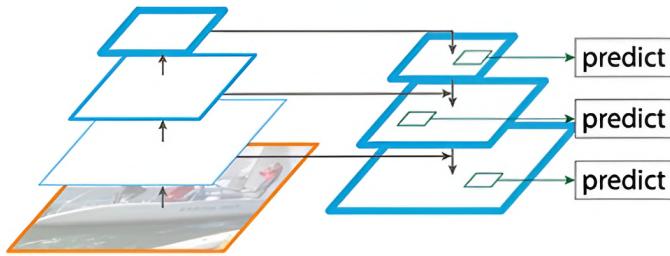


Figura 8.4.: El backbone que utiliza la estructura FPN está formado por rutas ascendentes y descendentes conectadas mediante conexiones de salto. Imagen extraída de [LDG⁺16].

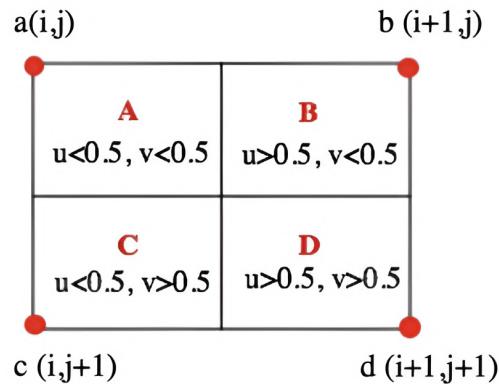


Figura 8.5.: Ejemplo de interpolación por vecino más cercano. Imagen elaborada por el autor.

do distintas resoluciones. Antes de comenzar a trabajar la Red de Propuestas de Regiones (RPN, por sus siglas en inglés), la cual calculará las regiones de interés (RoIs, por sus siglas en inglés), se selecciona el nivel adecuado de la FPN donde se procesará cada RoI, basado en el tamaño de esta. La fórmula utilizada para esta selección [Hui18b] es:

$$k = \left\lfloor k_0 + \log_2 \left(\frac{\sqrt{wh}}{224} \right) \right\rfloor,$$

donde w es el ancho y h es la altura del RoI, 224 representa el tamaño de preentrenamiento estándar en ImageNet, y k_0 es el nivel objetivo en el que se mapea un RoI de tamaño $w \cdot h = 224^2$. En Mask R-CNN, este último valor se fija en $k_0 = 4$, siguiendo la configuración de Faster R-CNN, que empleaba el cuarto nivel como único mapa de características de escala fija. De esta manera, los objetos grandes se procesan en mapas de baja resolución y los objetos pequeños en mapas de alta resolución, permitiendo que la RPN genere propuestas de RoIs con la escala más adecuada para cada objeto.

Después de seleccionar el mapa de características adecuado mediante la fórmula anterior, la RPN, que es una red completamente convolucional, comienza a trabajar. El mapa de características seleccionado se utiliza como entrada para la RPN, cuyo objetivo principal es obtener bounding boxes candidatas o RoIs con alta probabilidad de contener un objeto.

Primero, la RPN escanea la imagen mediante una ventana deslizante. El centro de la ventana deslizante en cada posición se denomina anchor. Para cada anchor, se define un conjunto de bounding boxes predefinidas (llamadas anchor boxes), que se emplean como referencia para predecir la ubicación de diferentes objetos. Normalmente, el número de anchor boxes generadas es de nueve [Bup18], cada una con distintas escalas y relaciones de aspecto (véase Figura 8.6).

Aunque los anchors están definidos sobre el mapa de características, hacen referencia a la imagen original. Una vez que se determinan todas las anchor boxes, la RPN los toma como entrada y produce dos salidas diferentes para cada uno de ellos. Por un lado, proporciona la probabilidad de que una anchor box contenga un objeto, usando una clasificación (con una función de pérdida binaria de entropía cruzada) que distingue entre dos categorías: fondo y primer plano (quedándose con las anchor boxes clasificadas en la categoría de primer plano). Por otro lado, la segunda salida es la regresión de las coordenadas de la anchor box. Es posible que la anchor box clasificada como primer plano no esté completamente centrada sobre el objeto que contiene, por lo que en esta etapa de regresión se estima un delta para ajustar la posición y el tamaño de la anchor box. Este delta representa un porcentaje de cambio en cuatro variables: x , y (coordenadas de la esquina superior izquierda de la anchor box), ancho y alto. Considerando las predicciones de la RPN, se seleccionan las anchor boxes con mayor probabilidad de contener un objeto y se utilizan los resultados de la regresión para refinárlas. Antes de que las anchor boxes refinadas se usen en la siguiente etapa, se aplica un proceso llamado supresión de no-máximos. Este proceso elimina las anchor boxes refinadas que se solapan con otras anchor boxes que tienen una mayor probabilidad de contener un objeto.

8.1.3. Capa de alineación de regiones de interés

Una vez que la RPN genera las anchor boxes refinadas con alta probabilidad de contener objetos, estas se utilizan como entrada para la siguiente etapa en la Capa de Alineación de Regiones de Interés (RoI Align Layer en inglés). En esta, Mask R-CNN realiza algunas mejoras con respecto a Faster R-CNN para adaptarla a la tarea de segmentación de instancias y mejorar el rendimiento. Primero, reemplaza la RoI Pooling Layer utilizada en Faster R-CNN por una RoI Align Layer. Además, añade una rama adicional para predecir una máscara para cada bounding box que contiene una instancia de una clase específica.

Las anchor boxes provenientes de la RPN tienen diferentes tamaños, lo cual genera mapas de características de distintos tamaños y tratar esto reduciría la eficiencia del modelo. Para abordar este problema, la RoI Align Layer transforma todas las anchor boxes al mismo tamaño.

En modelos anteriores como Faster R-CNN, se empleaba una RoI Pooling Layer, la cual recibía como entrada las regiones de interés generadas por la RPN. Esta capa seleccionaba

8. Métodos Propuestos

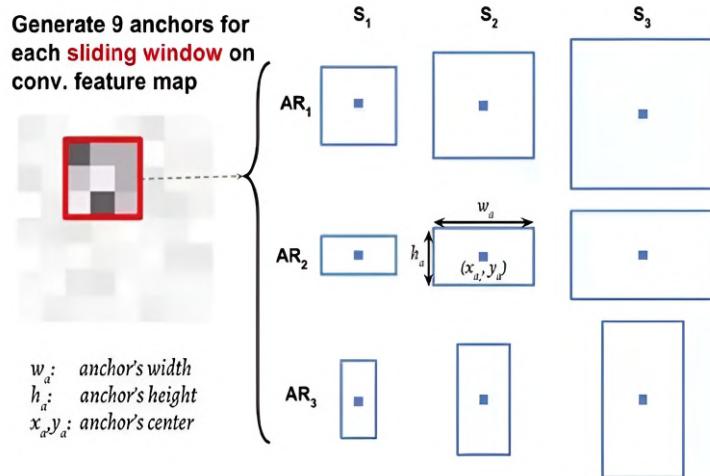


Figura 8.6.: Las distintas anchor boxes generadas tienen diferentes escalas y relaciones de aspecto. Imagen extraída de [Bup18].

secciones del mapa de características correspondientes a dichas RoIs y escalaba todas a un mismo tamaño, dividiendo las distintas RoIs en el mismo número de secciones o bins. Sin embargo, este proceso de cuantización no coincidía perfectamente con los límites de las RoIs ni del mapa de características, lo que provocaba desajustes y disminuía la precisión de las máscaras generadas.

Para resolver este inconveniente, Mask R-CNN utiliza una RoI Align Layer en lugar de RoI Pooling Layer. La RoI Align Layer elimina la necesidad de cuantización y genera bins de tamaño uniforme dentro de cada RoI mediante interpolación bilineal, lo cual permite que cada bin contenga puntos de muestra en su interior. Generalmente, se toman varios puntos en cada bin y luego se aplica una función de promedio o de máximo a estos puntos para asignar el resultado final.

El uso de una RoI Align Layer, en lugar de una RoI Pooling Layer, mejora significativamente la precisión de las máscaras predichas. La Figura 8.7 muestra la diferencia entre ambas capas.

8.1.4. Network head: Ramas de predicciones y segmentación

Una vez que todas las RoIs tienen las mismas dimensiones, estas se utilizan como entrada para el último módulo de Mask R-CNN. Este módulo final, conocido también como la network head, es donde se realizan las tareas finales de reconocimiento de bounding boxes, tanto de regresión como de clasificación (mediante capas completamente conectadas), y también la predicción de máscaras (mediante redes completamente convolucionales). Como se mencionó previamente, Mask R-CNN extiende a Faster R-CNN añadiendo una rama adicional para la predicción de máscaras.

Este módulo final es similar la RPN, con dos diferencias principales. En primer lugar, no

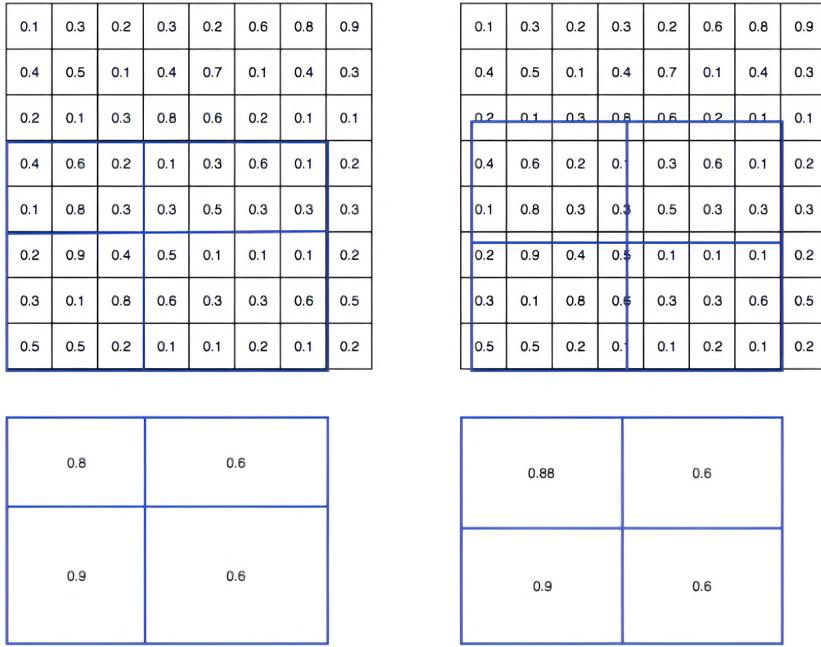


Figura 8.7.: Comparación entre RoI Pooling (izquierda) y RoI Align (derecha). Imagen extraída de [Hui18a].

emplea anchors para localizar las RoIs, ya que están determinadas, y fue la ROI Align Layer la encargada de ubicar las áreas relevantes correspondientes en los mapas de características. La segunda diferencia clave es la presencia de una rama adicional que, en paralelo a la regresión y clasificación de bounding boxes, realiza la predicción de máscaras para cada RoI. En la Figura 8.8 se observa un ejemplo de resultados obtenidos por Mask R-CNN para el conjunto de datos COCO [LMB⁺14].

8.1.5. Funciones de pérdida

Para entrenar Mask R-CNN y optimizar su rendimiento en la segmentación de instancias, se utilizan tres funciones de pérdida diferentes, cada una correspondiente a una de las ramas de la arquitectura: la clasificación, la regresión de bounding boxes y la segmentación de máscaras. La función de pérdida total, denotada como $\mathcal{L}_{\text{total}}$, es la combinación ponderada de estas tres pérdidas y se define de la siguiente manera:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{bbox}} + \mathcal{L}_{\text{mask}},$$

donde cada término en la ecuación representa una función de pérdida específica para una tarea dentro de Mask R-CNN, y su significado es el siguiente:

8. Métodos Propuestos



Figura 8.8.: Resultados de Mask R-CNN en el conjunto de test de COCO. Las máscaras se muestran en diferentes colores, e incluyen también la bounding box, la categoría y las puntuaciones de confianza. Imagen extraída de [HGDG17].

1. **Función de pérdida de clasificación (\mathcal{L}_{cls}):** Esta pérdida se utiliza en la rama de clasificación de Mask R-CNN para optimizar la precisión en la asignación de la clase correcta a cada región propuesta (RoI). Utiliza una entropía cruzada multiclas para medir la discrepancia entre la clase predicha y la clase real de cada RoI. En concreto, se define como:

$$\mathcal{L}_{cls} = -\frac{1}{N} \sum_{i=1}^N \log(p_i^c),$$

donde N es el número total de RoIs en el mini-batch utilizado para calcular la pérdida y p_i^c es la probabilidad predicha de que la i -ésima RoI pertenezca a la clase c , que es la clase verdadera asignada a esa RoI.

2. **Función de pérdida de regresión de bounding boxes (\mathcal{L}_{bbox}):** Esta pérdida se utiliza en la rama de regresión y permite ajustar la posición y el tamaño de las bounding boxes alrededor de cada objeto. Para ello, se emplea una pérdida de regresión L1 suavizada, que es una variante de la pérdida L1 que reduce la sensibilidad a errores de gran magnitud, ayudando así a la estabilidad y a la convergencia del modelo. Esta pérdida está formulada para minimizar la diferencia entre las coordenadas predichas de las bounding boxes y las coordenadas reales (ground truth). En particular:

$$\mathcal{L}_{bbox} = \sum_{i \in \{x,y,w,h\}} \text{smooth}_{L1}(\hat{t}_i - t_i),$$

donde \hat{t}_i representa las coordenadas predichas de la bounding box, t_i son las coordenadas de la bounding box real (ground truth), y smooth_{L1} es la función de pérdida L1

suavizada, definida como:

$$\text{smooth}_{L1}(x) = \begin{cases} 0.5x^2, & \text{si } |x| < 1 \\ |x| - 0.5, & \text{si } |x| \geq 1 \end{cases}.$$

3. **Función de pérdida de segmentación de máscaras ($\mathcal{L}_{\text{mask}}$)**: Esta pérdida se utiliza en la rama de segmentación de Mask R-CNN y optimiza la precisión en la predicción de las máscaras binarizadas de cada objeto detectado en las RoIs. La pérdida de máscara se define como una pérdida binaria de entropía cruzada promedio, que calcula la discrepancia entre la máscara predicha y la máscara real en cada píxel, aplicándose solo a la máscara correspondiente a la clase objetivo c de la RoI. Matemáticamente, se define como:

$$\mathcal{L}_{\text{mask}} = -\frac{1}{m^2} \sum_{i=1}^m \sum_{j=1}^m (y_{ij} \log(\hat{y}_{ij}^c) + (1 - y_{ij}) \log(1 - \hat{y}_{ij}^c)),$$

donde y_{ij} es el valor binario de la máscara real en el píxel (i, j) de la RoI (1 si el píxel pertenece al objeto y 0 si no), \hat{y}_{ij}^c es el valor predicho en el mismo píxel (i, j) en la máscara generada para la clase objetivo c , y m^2 es la resolución de la máscara. En este contexto, si C representa el número de categorías de objetos en la tarea de segmentación de instancias, entonces la salida de la máscara para cada RoI tiene una dimensión de $C \times m \times m$, donde $m \times m$ es el tamaño de la máscara. La red predice un valor binario para cada píxel en esta máscara y, posteriormente, determina si la máscara corresponde a una categoría específica. A la hora de medir el error, solo se considera la pérdida de la categoría correspondiente a la clase objetivo en cada RoI, ignorándose las pérdidas de otras categorías. Esto permite desacoplar la segmentación de máscaras de la clasificación de objetos. A diferencia de métodos de segmentación semántica, como las redes completamente convolucionales, donde cada píxel compite entre categorías para la clasificación, aquí cada máscara se calcula únicamente en función de su clase objetivo, evitando esta competencia entre categorías.

8.2. Grad-CAM

Grad-CAM (*Gradient-weighted Class Activation Mapping*) es una técnica de interpretabilidad post-hoc para redes neuronales convolucionales que permite identificar las áreas de una imagen que son más relevantes para la predicción de una clase específica. Desarrollado por Selvaraju et al. [SCD+17], Grad-CAM utiliza los gradientes asociados con las capas profundas de la red para generar mapas de calor que resaltan las regiones de la imagen de entrada que tienen una mayor influencia en la salida del modelo. A continuación, describimos detalladamente su funcionamiento.

8. Métodos Propuestos

8.2.1. Funcionamiento

Grad-CAM se basa en la observación de que las capas convolucionales profundas en una red neuronal convolucional capturan características visuales de alto nivel, mientras que las capas iniciales conservan información espacial detallada. Al analizar los gradientes de las activaciones en las capas convolucionales profundas respecto a la predicción de una clase específica, es posible visualizar qué regiones de la imagen fueron más determinantes en la toma de decisiones de la red. Esto resulta útil en aplicaciones donde se requiere una explicación visual de la decisión del modelo, como en la clasificación de imágenes y la detección de objetos.

El procedimiento seguido por Grad-CAM consta de varios pasos, los cuales se detallan a continuación:

1. **Activaciones de la capa convolucional seleccionada:** Sea f el modelo de red neuronal convolucional, y A^k el mapa de características de la capa convolucional seleccionada, donde k denota el índice del canal en dicha capa. Para una imagen de entrada x , el modelo produce una predicción para una clase específica, denotada como y^c , siendo c el índice de la clase de interés. El mapa de características A^k representa las activaciones de cada filtro convolucional en la capa seleccionada.
2. **Gradiente de la clase objetivo con respecto a las activaciones:** Para calcular la importancia de cada canal en la capa convolucional para la clase objetivo, Grad-CAM calcula los gradientes de la puntuación de la clase y^c con respecto a las activaciones A^k . Este gradiente indica el grado de influencia que tiene cada neurona en la capa convolucional sobre la predicción de la clase. La derivada se denota como $\frac{\partial y^c}{\partial A_{ij}^k}$, donde A_{ij}^k representa la activación del mapa de características A^k en la posición espacial (i, j) .
3. **Promedio global de los gradientes:** Para obtener un valor de importancia para cada canal k en la capa, se calcula el promedio global de los gradientes en todas las posiciones espaciales (i, j) . Este valor se usa como coeficiente ponderado de cada mapa de características A^k y se define como:

$$\alpha_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial A_{ij}^k},$$

donde Z es el número total de ubicaciones espaciales en el mapa de características A^k . Este promedio, α_k^c , representa la importancia del canal k para la predicción de la clase c . Los canales que tienen un mayor valor de α_k^c contribuyen más a la clase de interés c .

4. **Construcción del mapa de relevancia:** El mapa de relevancia para la clase c , denotado como $L_{\text{Grad-CAM}}^c$, se construye mediante una combinación lineal ponderada de los mapas de activación A^k con los coeficientes de importancia α_k^c :

$$L_{\text{Grad-CAM}}^c = \text{ReLU} \left(\sum_k \alpha_k^c A^k \right).$$

La función ReLU elimina valores negativos para garantizar que solo se mantengan las características que influyen positivamente en la predicción de la clase c .

5. **Interpretación del mapa de calor:** El mapa resultante $L_{\text{Grad-CAM}}^c$ se interpola al tamaño de la imagen de entrada y se superpone sobre esta para resaltar las áreas que tuvieron mayor influencia en la predicción. Las regiones con valores altos en el mapa de calor indican las áreas que el modelo consideró más relevantes para clasificar la imagen en la clase c .

En la Figura 8.9, se muestra un ejemplo del funcionamiento de Grad-CAM. La aplicación de Grad-CAM en problemas de clasificación proporciona una visualización intuitiva de la atención del modelo, facilitando la interpretabilidad en modelos complejos como las redes neuronales convolucionales.

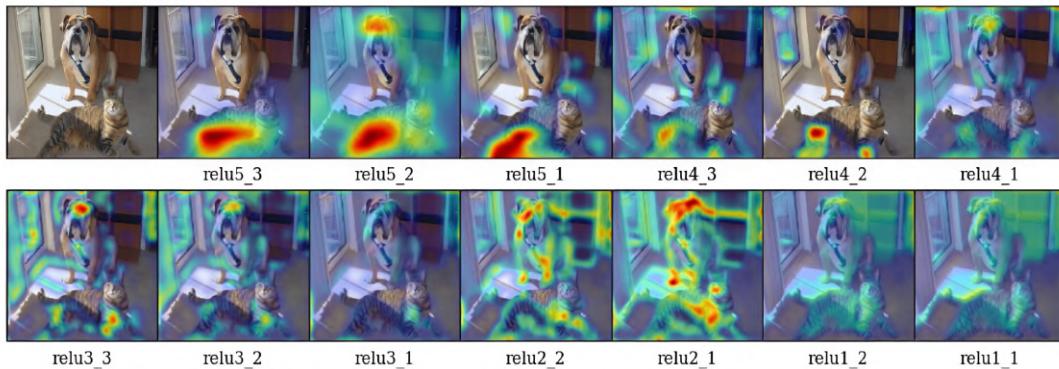


Figura 8.9.: Grad-CAM aplicado en diferentes capas convolucionales para la clase *tiger cat*.

Esta figura examina cómo las localizaciones varían cualitativamente al aplicar Grad-CAM en distintos mapas de características de una red neuronal convolucional (VGG16 [SZ14]). Observamos que las visualizaciones de mejor calidad se obtienen tras la capa convolucional más profunda de la red, mientras que la precisión de las localizaciones disminuye en capas más superficiales. Esto concuerda con lo comentado anteriormente, según lo cual las capas convolucionales más profundas capturan conceptos semánticos más complejos. Imagen extraída de [SCD⁺17].

8.2.2. Ventajas y limitaciones

Grad-CAM ofrece una serie de ventajas que lo convierten en una herramienta poderosa para la interpretabilidad de modelos de redes neuronales convolucionales:

8. Métodos Propuestos

- Permite interpretar las predicciones de una red neuronal convolucional mediante mapas de calor, facilitando la explicación visual de las decisiones del modelo.
- Al emplear gradientes, puede aplicarse en redes neuronales profundas sin necesidad de modificar su estructura ni entrenarlas nuevamente.
- Es adaptable a diferentes arquitecturas y problemas, incluyendo clasificación.

Sin embargo, también presenta algunas limitaciones considerables:

- Puede tener dificultades en la interpretación de modelos complejos cuando las características visuales de interés están en capas muy profundas.
- La precisión del mapa de calor puede verse afectada por las características espaciales limitadas en las capas de mayor abstracción.

9. Implementación

9.1. Diseño del software

9.2. Entorno de ejecución

10. Experimentación

10.1. Datos empleados

10.2. Protocolo de validación experimental

10.3. Métricas empleadas

10.4. Experimentos realizados

10.5. Resultados obtenidos

10.6. Discusión de los resultados obtenidos

11. Conclusiones y Trabajos Futuros

Bibliografía

- [AAG93] S. T. Ali, J. P. Antoine, y J. P. Gazeau. Continuous frames in hilbert spaces. *Annals of Physics*, 222(1):1–37, 1993.
- [AB18] A. Adadi y M. Berrada. Peeking inside the black-box: A survey on explainable artificial intelligence (xai). *IEEE Access*, 6:52138–52160, 2018.
- [AM14] J. Andén y S. Mallat. Deep scattering spectrum. *IEEE Transactions on Signal Processing*, 62(16):4114–4128, 2014.
- [AMVAo8] J. P. Antoine, R. Murenzi, P. Vandergheynst, y S. T. Ali. Two-dimensional wavelets and their relatives. *Cambridge University Press*, 2008.
- [APYS23] S. Ao, M. Pingchuan, Y. Yuanyuan, y W. Shuai. Explain any concept: Segment anything meets concept-based explanation. *arXiv preprint arXiv:2305.10289*, 2023.
- [ASJ⁺21] P. P. Angelov, E. A. Soares, R. Jiang, N. I. Arnold, y P. M. Atkinson. Explainable artificial intelligence: an analytical review. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 11(5), 2021.
- [BBM⁺15] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, y W. Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7), 2015.
- [BCV13] Y. Bengio, A. Courville, y P. Vincent. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828, 2013.
- [Ben12] Y. Bengio. Deep learning of representations for unsupervised and transfer learning. *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, 2012.
- [Bis94] C. M. Bishop. Neural networks and their applications. *Review of Scientific Instruments*, 65(6):1803–1832, 1994.
- [BM13] J. Bruna y S. Mallat. Invariant scattering convolution networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1872–1886, 2013.
- [Bru12] J. Bruna. Operators commuting with diffeomorphisms. Technical report, CMAP Tech. Report, 2012.
- [BU16] M. Bai y R. Urtasun. Deep watershed transform for instance segmentation. *CoRR*, abs/1611.08303, 2016.
- [Bup18] C. Bupe. How does the region proposal network (rpn) in faster r-cnn work? <https://www.quora.com/How-does-the-region-proposal-network-RPN-in-Faster-R-CNN-work>, 2018.
- [CCG91] S. Chen, C. Cowan, y P. M. Grant. Orthogonal least squares learning algorithm for radial basis function networks. *IEEE Transactions on Neural Networks*, 2(2):302–309, 1991.
- [Chi17] S. Chilamkurthy. A 2017 guide to semantic segmentation with deep learning. <http://blog.qure.ai/notes/semantic-segmentation-deep-learning-review>, 2017.
- [DHL⁺16] J. Dai, K. He, Y. Li, S. Ren, y J. Sun. Instance-sensitive fully convolutional networks. *CoRR*, abs/1603.08678, 2016.
- [DLHS16] J. Dai, Y. C. Li, K. He, y J. Sun. R-fcn: Object detection via region-based fully convolutional networks. *Advances in Neural Information Processing Systems (NIPS)*, 2016.

Bibliografía

- [DSZ⁺18] Z. Deng, H. Sun, S. Zhou, J. Zhao, Y. Wang, y H. Wang. Multi-scale object detection in remote sensing imagery with convolutional neural networks. *ISPRS Journal of Photogrammetry and Remote Sensing*, 145:3–15, 2018.
- [DVC⁺16] M. Drozdzal, E. Vorontsov, G. Chartrand, S. Kadouri, y C. Pal. The importance of skip connections in biomedical image segmentation. *CoRR*, abs/1608.04117, 2016.
- [DVK17] F. Doshi-Velez y B. Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.
- [EEG⁺15] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, y A. Zisserman. The pascal visual object classes challenge: A retrospective. <http://host.robots.ox.ac.uk/pascal/VOC/>, 2015.
- [ESTA13] D. Erhan, C. Szegedy, A. Toshev, y D. Anguelov. Scalable object detection using deep neural networks. *CoRR*, abs/1312.2249, 2013.
- [GB10] X. Glorot y Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. *Proc. of International Conference on Artificial Intelligence and Statistics (AISTATS)*, páginas 249–256, 2010.
- [GBB11] X. Glorot, A. Bordes, y Y. Bengio. Deep sparse rectifier neural networks. *Proc. of International Conference on Artificial Intelligence and Statistics (AISTATS)*, páginas 315–323, 2011.
- [GBC16] I. Goodfellow, Y. Bengio, y A. Courville. *Deep Learning*. MIT Press, 2016.
- [GDDM13] R. B. Girshick, J. Donahue, T. Darrell, y J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013.
- [Gir15] R. B. Girshick. Fast r-cnn. *CoRR*, abs/1504.08083, 2015.
- [HAE16] M. Huh, P. Agrawal, y A. A. Efros. What makes imagenet good for transfer learning? *arXiv preprint arXiv:1608.08614*, 2016.
- [HAGM14a] B. Hariharan, P. Arbeláez, R. B. Girshick, y J. Malik. Hypercolumns for object segmentation and fine-grained localization. *CoRR*, abs/1411.5752, 2014.
- [HAGM14b] B. Hariharan, P. Arbeláez, R. B. Girshick, y J. Malik. Simultaneous detection and segmentation. *CoRR*, abs/1407.1808, 2014.
- [HDO⁺98] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, y B. Schölkopf. Support vector machines. *IEEE Intelligent Systems and their Applications*, 13(4):18–28, 1998.
- [HGDG17] K. He, G. Gkioxari, P. Dollár, y R. Girshick. Mask r-cnn. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, páginas 2961–2969, 2017.
- [HL06] F. J. Huang y Y. LeCun. Large-scale learning with svm and convolutional nets for generic object categorization. *Proc. of IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, páginas 284–291, 2006.
- [Hui18a] J. Hui. Image segmentation with mask r-cnn. https://medium.com/@jonathan_hui/image-segmentation-with-mask-r-cnn-ebe6d793272, 2018.
- [Hui18b] J. Hui. Understanding feature pyramid networks for object detection (fpn). https://medium.com/@jonathan_hui/understanding-feature-pyramid-networks-for-object-detection-fpn-45b227b9106c, 2018.
- [HZRS14] K. He, X. Zhang, S. Ren, y J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *CoRR*, abs/1406.4729, 2014.
- [HZRS15] K. He, X. Zhang, S. Ren, y J. Sun. Deep residual learning for image recognition. *Proc. of IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, páginas 770–778, 2015.
- [Hö71] L. Hörmander. Fourier integral operators. i. *Acta Mathematica*, 127(1):79–183, 1971.

- [IS15] S. Ioffe y C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *International Conference on Machine Learning*, páginas 448–456, 2015.
- [JDV⁺16] S. Jégou, M. Drozdzal, D. Vázquez, A. Romero, y Y. Bengio. The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. *CoRR*, abs/1611.09326, 2016.
- [JKRL09] K. Jarrett, K. Kavukcuoglu, M. Ranzato, y Y. LeCun. What is the best multi-stage architecture for object recognition? *Proc. of IEEE International Conference on Computer Vision (ICCV)*, páginas 2146–2153, 2009.
- [Jor18a] J. Jordan. An overview of object detection: one-stage methods. <https://www.jeremyjordan.me/object-detection-one-stage/>, 2018.
- [Jor18b] J. Jordan. An overview of semantic image segmentation. <https://www.jeremyjordan.me/semantic-segmentation/>, 2018.
- [Jor24] P. Jorgensen. Haar Wavelet. <https://homepage.divms.uiowa.edu/~jorgen/Haar.html>, 2024.
- [KA18] E. Kauderer-Abrams. Quantifying translation-invariance in convolutional neural networks. *CoRR*, abs/1801.01450, 2018.
- [Kai94] G. Kaiser. *A Friendly Guide to Wavelets*. Birkhäuser, 1994.
- [Kan15] T. Kang. Using neural networks for image classification. *San Jose State University*, 2015.
- [KLA⁺16] A. Kirillov, E. Levinkov, B. Andres, B. Savchynskyy, y C. Rother. Instancecut: From edges to instances with multicut. *CoRR*, abs/1611.08272, 2016.
- [Koz08] L. Kozma. K nearest neighbors algorithm (knn). Technical report, Helsinki University of Technology, 2008.
- [KSH17] A. Krizhevsky, I. Sutskever, y G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, May 2017.
- [KSI⁺24] M. Kowal, M. Siam, M. A. Islam, N. D. B. Bruce, R. P. Wildes, y K. G. Derpanis. Quantifying and learning static vs. dynamic information in deep spatiotemporal networks. *arXiv preprint arXiv:2211.01783v2*, 2024.
- [KSL19] S. Kornblith, J. Shlens, y Q. V. Le. Do better imagenet models transfer better? *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, páginas 2661–2671, 2019.
- [LAE⁺15] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, y A. C. Berg. Ssd: Single shot multibox detector. *CoRR*, abs/1512.02325, 2015.
- [LBD⁺90] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, y L. Jackel. Handwritten digit recognition with a back-propagation network. *Proc. of International Conference on Neural Information Processing Systems (NIPS)*, páginas 396–404, 1990.
- [LBH15] Y. LeCun, Y. Bengio, y G. Hinton. Deep learning. *Nature*, 521:436–444, 2015.
- [LC98] Y. LeCun y C. Cortes. The mnist database of handwritten digits. <http://yann.lecun.com/exdb/mnist/>, 1998.
- [LCY13] M. Lin, Q. Chen, y S. Yan. Network in network. *CoRR*, abs/1312.4400, 2013.
- [LDG⁺16] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, y S. J. Belongie. Feature pyramid networks for object detection. *CoRR*, abs/1612.03144, 2016.
- [Lee96] T. Lee. Image representation using 2d gabor wavelets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(10):959–971, 1996.

Bibliografía

- [Liu18] Yanfeng Liu. The confusing metrics of ap and map for object detection / instance segmentation. <https://yanfengliux.medium.com/the-confusing-metrics-of-ap-and-map-for-object-detection-3113ba0386ef>, 2018.
- [LJFU17] S. Liu, J. Jia, S. Fidler, y R. Urtasun. Sgn: Sequential grouping networks for instance segmentation. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, páginas 3516–3524, October 2017.
- [LKF10] Y. LeCun, K. Kavukcuoglu, y C. Farabet. Convolutional networks and applications in vision. *Proc. of IEEE International Symposium on Circuits and Systems (ISCAS)*, páginas 253–256, 2010.
- [LL17] S. M. Lundberg y S. I. Lee. A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, páginas 4765–4774, 2017.
- [LMB⁺14] T. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, y P. Dollár. Microsoft coco: Common objects in context. *CoRR*, abs/1405.0312, 2014.
- [Low04] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60, 2004.
- [LOW⁺19] L. Liu, W. Ouyang, X. Wang, P. W. Fieguth, J. Chen, X. Liu, y M. Pietikäinen. Deep learning for generic object detection: A survey. *arXiv preprint arXiv:1809.02165*, 2019.
- [LQD⁺16] Y. Li, H. Qi, J. Dai, X. Ji, y Y. Wei. Fully convolutional instance-aware semantic segmentation. *CoRR*, abs/1611.07709, 2016.
- [LSD14] J. Long, E. Shelhamer, y T. Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014.
- [LYP16] S. Lim, S. Young, y R. Patton. An analysis of image storage systems for scalable training of deep neural networks. *ResearchGate*, 2016.
- [Malo8] S. Mallat. *A Wavelet Tour of Signal Processing*. Elsevier, 3rd ed. edición, 2008.
- [Mal12] S. Mallat. Group invariant scattering. *Comm. Pure Appl. Math.*, 65(10):1331–1398, 2012.
- [Mat24a] MathWorks. Continuous wavelet transform and scale-based analysis. <https://www.mathworks.com/help/wavelet/gs/continuous-wavelet-transform-and-scale-based-analysis.html>, 2024.
- [Mat24b] MathWorks. Redes neuronales convolucionales. <https://es.mathworks.com/discovery/convolutional-neural-network.html>, 2024.
- [MDH11] A. Mohamed, G. E. Dahl, y G. Hinton. Acoustic modeling using deep belief networks. *IEEE Trans. Audio, Speech, and Language Process.*, 20:14–22, Jan. 2011.
- [Mit97] T. M. Mitchell. *Machine Learning*, volumen 1. McGraw-Hill, New York, 1997.
- [ML06] J. Mutch y D. Lowe. Multiclass object recognition with sparse, localized features. *Proc. of IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, páginas 11–18, 2006.
- [MLB⁺17] G. Montavon, S. Lapuschkin, A. Binder, W. Samek, y K.-R. Müller. Explaining nonlinear classification decisions with deep taylor decomposition. *Pattern Recognition*, 65:211–222, 2017.
- [MS24] Z. Mousses y O. Sedat. Improving the explain-any-concept by introducing nonlinearity to the trainable surrogate model. *arXiv preprint arXiv:2405.11837*, 2024.
- [MSM18] G. Montavon, W. Samek, y K. R. Müller. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 73:1–15, 2018.
- [NH10] V. Nair y G. Hinton. Rectified linear units improve restricted boltzmann machines. *Proc. of International Conference on Machine Learning (ICML)*, páginas 807–814, 2010.

- [NRD15] M. Najibi, M. Rastegari, y L. S. Davis. G-cnn: An iterative grid based object detector. *CoRR*, abs/1512.07729, 2015.
- [Onl24] S. Online. Activation functions with real-life analogy and python code. <https://www.shiksha.com/online-courses/articles/activation-functions-with-real-life-analogy-and-python-code/>, 2024.
- [Oua17] A. Ouaknine. Review of deep learning algorithms for image classification. <https://medium.com/zylapp/review-of-deep-learning-algorithms-for-image-classification-5fdca4a05e2>, 2017.
- [PAB⁺15] J. Pont-Tuset, P. Arbeláez, J. T. Barron, F. Marqués, y J. Malik. Multiscale combinatorial grouping for image segmentation and object proposal generation. *CoRR*, abs/1503.00848, 2015.
- [PCD08] N. Pinto, D. Cox, y J. DiCarlo. Why is real-world visual object recognition hard. *PLoS Computational Biology*, 4(1):151–156, 2008.
- [PCD15] P. H. O. Pinheiro, R. Collobert, y P. Dollár. Learning to segment object candidates. *CoRR*, abs/1506.06204, 2015.
- [Pin17] J. Pinto. Masknet: An instance segmentation algorithm. *Chalmers University*, 2017.
- [PY10] Sinno Jialin Pan y Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.
- [RDGF15] J. Redmon, S. K. Divvala, R. B. Girshick, y A. Farhadi. You Only Look Once: Unified, Real-Time Object Detection. *CoRR*, abs/1506.02640, 2015.
- [RFB15] O. Ronneberger, P. Fischer, y T. Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.
- [RHBL07] M. A. Ranzato, F. J. Huang, Y. Boureau, y Y. LeCun. Unsupervised learning of invariant feature hierarchies with applications to object recognition. *Proc. of IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, páginas 1–8, 2007.
- [RHGS15] S. Ren, K. He, R. B. Girshick, y J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.
- [RMKH24] B. Rajeswari, R. J. Mani, D.V.T. Kumar, y K.L.V.V. Harshith. Review of semantic segmentation by using deep learning methods. *Proceedings of the 2024 International Conference on Social and Sustainable Innovations in Technology and Engineering (SASI-ITE 2024)*, páginas 272–277, 2024.
- [RN02] S. Russell y P. Norvig. *Artificial Intelligence: A Modern Approach*, volumen 90. Prentice Hall, 2002.
- [Roh18] G. Rohith. R-cnn, fast r-cnn, faster r-cnn, yolo — object detection algorithms. <https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e>, 2018.
- [Ros88] A. Rosenfeld. Computer vision: basic principles. *Proceedings of the IEEE*, 76(8):863–868, 1988.
- [RPCLo6] M. Ranzato, C. Poultney, S. Chopra, y Y. LeCun. Efficient learning of sparse representations with an energy-based model. *Proc. of International Conference on Neural Information Processing Systems (NIPS)*, páginas 1137–1144, 2006.
- [RSG16] M. T. Ribeiro, S. Singh, y C. Guestrin. "why should i trust you?": Explaining the predictions of any classifier. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, páginas 1135–1144, 2016.
- [Rud91] W. Rudin. *Functional Analysis*. McGraw.Hill, 2nd ed. edición, 1991.

Bibliografía

- [Rud16] S. Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint, abs/1609.04747*, 2016.
- [Sam59] A. L. Samuel. Machine learning. *The Technology Review*, 62(1):42–45, 1959.
- [SCD⁺17] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, y D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, páginas 618–626, 2017.
- [Sco24] Scopus. Scopus - Abstract and Citation Database. <https://www.scopus.com>, 2024.
- [SDBR14] J. T. Springenberg, A. Dosovitskiy, T. Brox, y M. Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.
- [SHII20] N. Sabri, H. N. A. Hamed, Z. Ibrahim, y K. Ibrahim. A comparison between average and max-pooling in convolutional neural network for scoliosis classification. *International Journal of Advanced Trends in Computer Science and Engineering*, 9(1.4):689–696, 2020.
- [Sif14] L. Sifre. Rigid-motion scattering for texture classification. *Centre de Mathématiques Appliquées, École Polytechnique Paris-Saclay*, 2014.
- [SKY23] SKY ENGINE AI. What is Mask R-CNN? <https://skyengine.ai/se/skyengine-blog/119-what-is-mask-r-cnn>, 2023.
- [SLJ⁺14] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, y A. Rabinovich. Going deeper with convolutions. *CoRR, abs/1409.4842*, 2014.
- [Sor18] M. Sorsater. Active learning for road segmentation using convolutional neural networks. *Linkoping University*, 2018.
- [SRC24] J. Stodt, C. Reich, y N. Clarke. Unified intersection over union for explainable artificial intelligence. *Lecture Notes in Networks and Systems, 2024 Intelligent Systems Conference, IntelliSys 2023*, 823:758–770, 2024.
- [STY17] M. Sundararajan, A. Taly, y Q. Yan. Axiomatic attribution for deep networks. *International Conference on Machine Learning*, páginas 3319–3328, 2017.
- [SVZ13] K. Simonyan, A. Vedaldi, y A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.
- [SWP05] T. Serre, L. Wolf, y T. Poggio. Object recognition with features inspired by visual cortex. *Proc. of IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, páginas 994–1000, 2005.
- [SZ14] K. Simonyan y A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR, abs/1409.1556*, 2014.
- [Uni20] Stanford University. Cs231n: Convolutional neural networks for visual recognition. <https://cs231n.stanford.edu/2020/syllabus.html>, 2020.
- [USGS13] J. Uijlings, K. E. A. Van De Sande, T. Gevers, y A. W. M. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104:154–171, September 2013.
- [Vai93] P. P. Vaidyanathan. *Multirate Systems and Filter Banks*. Prentice Hall, 1993.
- [WB18] T. Wiatowski y H. Bölcskei. A mathematical theory of deep convolutional neural networks for feature extraction. *IEEE Transactions on Information Theory*, 63:1845–1866, 2018.
- [Wik24a] Wikipedia. Gabor wavelet – Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/wiki/Gabor_wavelet, 2024.
- [Wik24b] Wikipedia. Ondícula de Haar — Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Ond%C3%ADcula_de_Haar, 2024.
- [Wik24c] Wikipedia. Perceptrón multicapa — wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Perceptr%C3%B3n_multicapa, 2024.

- [WTS⁺16] T. Wiatowski, M. Tschannen, A. Stanić, P. Grohs, y H. Bölcskei. Discrete deep feature extraction: A theory and new architectures. *Proc. of International Conference on Machine Learning (ICML)*, páginas 2149–2158, 2016.
- [XJ21] A. I. Xavier y J. Jeng. Mask-gradcam: Object identification and localization of visual presentation for deep convolutional network. *2021 6th International Conference on Inventive Computation Technologies (ICICT)*, páginas 1171–1178, January 2021.
- [XVM⁺21] A. I. Xavier, C. Villavicencio, J. J. Macrohon, J. Jeng, y J. Hsieh. Object identification and localization using grad-cam++ with mask regional convolution neural network. *Electronics*, 10(13):1541, 2021.
- [XVM⁺22] A. I. Xavier, C. Villavicencio, J. J. Macrohon, J. Jeng, y J. Hsieh. Object detection via gradient-based mask r-cnn using machine learning algorithms. *Machines*, 10(5):340, 2022.
- [YCBL14] J. Yosinski, J. Clune, Y. Bengio, y H. Lipson. How transferable are features in deep neural networks? *Proceedings of the 27th International Conference on Neural Information Processing Systems*, página 3320–3328, 2014.
- [YPL⁺15] D. Yoo, S. Park, J. Lee, A. S. Paek, y I. Kweon. Attentionnet: Aggregating weak directions for accurate object detection. *CoRR*, abs/1506.07704, 2015.
- [ZKS⁺19] Z. Zhao, A. Kleinhans, G. Sandhu, I. Patel, y K. P. Unnikrishnan. Capsule networks with max-min normalization. *arXiv preprint arXiv:1903.09662*, 2019.
- [ZZXW18] Z. Zhao, P. Zheng, S. Xu, y X. Wu. Object detection with deep learning: A review. *CoRR*, abs/1807.05511, 2018.