



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 para la Segmentación de Instancias

Presentado por:

Juan Manuel Rodríguez Gómez

Tutores:

Francisco Javier Merí de la Maza

Departamento de Análisis Matemático

Pablo Mesejo Santiago

Departamento de Ciencias de la Computación e Inteligencia Artificial

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 para la
Segmentación de Instancias

Juan Manuel Rodríguez Gómez

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

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

Responsables de tutorización

Francisco Javier Merí de la Maza
Departamento de Análisis Matemático

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

Marilyn Bello García
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

Facultad de Ciencias
Universidad de Granada

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 4 de junio de 2025.

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 profundamente a mis tutores, Francisco Javier Merí de la Maza, Pablo Mesejo Santiago y Marilyn Bello García, cuya implicación y dedicación constante han sido clave para el desarrollo de este trabajo. Vuestro acompañamiento cercano, compromiso y guía académica han sido fundamentales desde el inicio hasta la finalización de este trabajo. A los tres, gracias por compartir generosamente vuestro conocimiento y por orientarme con tanta claridad y cercanía en cada etapa del proceso.

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 la familia de mi pareja, por su cercanía, cariño y apoyo constante. Gracias por hacerme sentir siempre bienvenido y por brindarme un entorno de tranquilidad y confianza que ha sido muy valioso a lo largo de este camino.

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: Redes Neuronales Convolucionales, Modelización Matemática, Ondículas, Visión por Computador, Aprendizaje Profundo, Segmentación de Instancias, Inteligencia Artificial Explicable.

Este Trabajo de Fin de Grado combina dos enfoques complementarios: por un lado, el desarrollo de una modelización matemática general de las redes neuronales convolucionales, y por otro, la adaptación y evaluación de un modelo de segmentación de instancias en imágenes dentales mediante técnicas de inteligencia artificial explicable.

En la parte matemática, se extiende el análisis de Mallat sobre redes de dispersión [Mal12] mediante el marco propuesto por Wiatowski y Bölskei [WB18], que permite caracterizar redes neuronales convolucionales profundas mediante marcos semi-discretos generales, funciones de activación y operadores de *pooling* Lipschitz-continuos. Bajo esta formulación, se demuestra la invarianza vertical a traslaciones, es decir, la creciente robustez del extracto de características ante desplazamientos espaciales conforme se profundiza en la red.

En la parte informática, se aplica la arquitectura Mask R-CNN [HGDG17] a la segmentación de instancias dentales en radiografías panorámicas. Se adapta un modelo preentrenado en COCO al dominio específico mediante ajuste fino y se implementa la técnica de explicabilidad Grad-CAM [SCD⁺17] siguiendo las adaptaciones propuestas por Inbaraj et al. [XJ²¹, XVM⁺₂₁, XVM⁺₂₂], con el objetivo de facilitar la comprensión de sus predicciones. A esta propuesta la hemos denominado DentCAM-RCNN. La evaluación incluye métricas de detección (Precision, Recall, F1-Score, AP@50), segmentación (DSC, IoU, HD) y explicabilidad (Deletion, Insertion, Stability, IoU_{Grad-CAM}), desglosadas por categoría dental.

DentCAM-RCNN alcanza un F1-Score del 96.04 %, una precisión del 97.36 %, un DSC del 87.76 % y un IoU del 79.23 % en el conjunto de test, superando ampliamente tanto al modelo base como a métodos clásicos de segmentación no supervisados. Estos resultados se consolidan mediante una comparación cuantitativa con estudios recientes que utilizan la arquitectura Mask R-CNN en tareas similares, donde el modelo propuesto destaca por su rendimiento robusto, incluso con un volumen intermedio de datos.

Los resultados obtenidos refuerzan el valor de las redes neuronales convolucionales ajustadas al dominio y complementadas con técnicas de explicabilidad *post-hoc* como Grad-CAM, tanto para alcanzar resultados precisos como para dotar de explicabilidad a modelos aplicables en el análisis automatizado de radiografías dentales.

Summary

KEYWORDS: Convolutional Neural Networks, Mathematical Modeling, Wavelets, Computer Vision, Deep Learning, Instance Segmentation, Explainable Artificial Intelligence.

This Bachelor's Thesis combines two complementary approaches: on the one hand, the development of a general mathematical modeling of convolutional neural networks, and on the other, the adaptation and evaluation of an instance segmentation model on dental images using explainable artificial intelligence techniques.

In the mathematical part, Mallat's analysis of scattering networks [Mal12] is extended through the framework proposed by Wiatowski and Bölcskei [WB18], which allows deep convolutional neural networks to be characterized via general semi-discrete frames, activation functions, and Lipschitz-continuous pooling operators. Under this formulation, vertical translation invariance is demonstrated, meaning the increasing robustness of the feature extractor to spatial displacements as the network depth increases.

In the computational part, the Mask R-CNN architecture [HGDG17] is applied to the task of instance segmentation in panoramic dental radiographs. A model pretrained on COCO is fine-tuned to the specific domain, and the explainability technique Grad-CAM [SCD⁺17] is implemented following the adaptations proposed by Inbaraj et al. [XJ²¹, XVM⁺₂₁, XVM⁺₂₂], with the aim of facilitating the understanding of its predictions. We have named this approach DentCAM-RCNN. The evaluation includes detection metrics (Precision, Recall, F1-Score, AP@50), segmentation metrics (DSC, IoU, HD), and explainability metrics (Deletion, Insertion, Stability, IoU_{Grad-CAM}), all broken down by dental category.

DentCAM-RCNN achieves an F1-Score of 96.04%, a precision of 97.36%, a DSC of 87.76%, and an IoU of 79.23% on the test set, significantly outperforming both the base model and classical unsupervised segmentation methods. These results are further supported by a quantitative comparison with recent studies using the Mask R-CNN architecture for similar tasks, where the proposed model stands out for its robust performance, even with an intermediate data volume.

The results obtained highlight the value of convolutional neural networks tailored to the domain and enhanced with *post-hoc* explainability techniques such as Grad-CAM, both for achieving accurate results and for providing interpretability to models applicable to the automated analysis of dental radiographs.

Índice general

Agradecimientos	v
Resumen	VII
Summary	IX
Índice de figuras	XV
Índice de tablas	XVII
I. Análisis de Redes Convolucionales	1
1. Introducción	3
2. Teoría de Mallat sobre Redes de Dispersión	7
2.1. Evolución matemática: De Fourier a Littlewood-Paley	7
2.1.1. La transformada de Fourier y sus limitaciones	7
2.1.2. Ondículas como alternativa a la transformada de Fourier	9
2.1.3. La transformada de Littlewood-Paley	13
2.1.4. Convenciones para futuras secciones	17
2.2. El operador de dispersión sobre un camino ordenado	18
2.2.1. Obtención de coeficientes invariantes por traslaciones	18
2.2.2. El operador módulo	19
2.2.3. El propagador de dispersión	21
2.2.4. La transformada de dispersión	22
2.3. Propagación de la dispersión y conservación de la norma	23
2.3.1. Proceso iterativo del propagador de la dispersión	23
2.3.2. Comparativa con una red neuronal convolucional	24
2.3.3. No-expansividad en la distancia entre funciones	25
2.3.4. Conservación de la norma	26
2.4. Invarianza horizontal por traslaciones	31
2.4.1. No-expansividad en conjuntos de caminos	31
2.4.2. Demostración de la invarianza por traslaciones	33
2.5. Invarianza frente a pequeñas deformaciones	37
2.5.1. Cotas superiores en comutadores de dispersión	37
2.5.2. Demostración de la invarianza frente a pequeñas deformaciones	39
3. Generalización de la Teoría de Mallat sobre Redes de Dispersión	43
3.1. Resumen de la teoría de Mallat sobre redes de dispersión	43
3.1.1. Definición del vector de características en redes de dispersión	43
3.1.2. Marcos semi-discretos	44

Índice general

3.1.3. Relación entre la arquitectura de redes de dispersión y las características extraídas	45
3.2. Construcción del extractor de características convolucional general	46
3.2.1. Diferencias y similitudes con las redes de dispersión	46
3.2.2. Operadores de <i>pooling</i>	48
3.3. Invarianza vertical por traslaciones	55
4. Conclusiones y Trabajos Futuros	63
II. Técnicas de Explicabilidad para Arquitecturas Neuronales Profundas de Segmentación de Instancias	65
5. Introducción	67
5.1. Descripción del problema y motivación	67
5.2. Objetivos	69
5.3. Planificación y estimación de costes	70
6. Fundamentos Teóricos	73
6.1. Aprendizaje Automático	73
6.1.1. Aprendizaje Supervisado	74
6.1.2. Aprendizaje No Supervisado	76
6.1.3. Aprendizaje Profundo	76
6.1.4. Redes neuronales	76
6.2. Redes neuronales convolucionales	79
6.2.1. Capa convolucional	80
6.2.2. Capa de <i>pooling</i>	82
6.2.3. Capa totalmente conectada	82
6.2.4. Batch normalization	83
6.2.5. Proceso de entrenamiento	84
6.2.6. <i>Fine-tuning</i>	84
6.3. Visión por Computador	85
6.3.1. Clasificación de imágenes	86
6.3.2. Detección de objetos	87
6.3.3. Segmentación semántica	88
6.3.4. Segmentación de instancias	89
6.4. Arquitecturas R-CNN	90
6.4.1. Arquitectura R-CNN	90
6.4.2. Arquitectura Fast R-CNN	91
6.4.3. Arquitectura Faster R-CNN	92
6.5. Inteligencia Artificial Explicable	93
6.5.1. Conceptos fundamentales	93
6.5.2. Tipos de explicabilidad	94
6.5.3. Propiedades deseables	95
6.5.4. Técnicas de explicabilidad <i>post-hoc</i> en redes neuronales convolucionales	96
7. Estado del Arte	99
7.1. Segmentación de imágenes dentales	99

7.2. Técnicas de explicabilidad aplicadas a Mask R-CNN	102
8. Métodos Empleados	107
8.1. Mask R-CNN	107
8.1.1. <i>Backbone</i> : Red de extracción de características	107
8.1.2. Red de propuestas de regiones	111
8.1.3. Capa de alineación de regiones de interés	112
8.1.4. <i>Network head</i> : Ramas de predicciones y segmentación	113
8.1.5. Funciones de pérdida	114
8.2. Grad-CAM	115
8.2.1. Funcionamiento	116
8.2.2. Ventajas y limitaciones	118
8.3. Adaptación de Grad-CAM a Mask R-CNN	118
9. Implementación	121
9.1. Diseño del software	121
9.2. Entorno de ejecución	123
10. Experimentación	127
10.1. Datos empleados	127
10.2. Configuración de hiperparámetros	133
10.3. Métricas empleadas para la evaluación	133
10.3.1. Métricas de detección	133
10.3.2. Métricas de segmentación	134
10.3.3. Métricas de explicabilidad	135
10.4. Hipótesis iniciales	138
10.5. Evaluación y análisis de resultados	138
10.5.1. Comparación entre modelo base y modelo ajustado	138
10.5.2. Comparación de DentCAM-RCNN con métodos clásicos de segmentación	147
10.5.3. Comparación de DentCAM-RCNN con estudios recientes de segmentación de instancias dentales en 2D	148
11. Conclusiones y Trabajos Futuros	153
A. Apéndice: Evaluación detallada del modelo base	155
B. Apéndice: Evaluación detallada del modelo ajustado	159
Bibliografía	163

Índice de figuras

1.1.	Dígitos MNIST – Ejemplo de invarianza por traslaciones	3
1.2.	Dígitos MNIST – Ejemplo de invarianza frente a pequeñas deformaciones	4
1.3.	Dígitos MNIST – Ejemplos de fallos de predicción causados por deformaciones en los dígitos	5
2.1.	Función base de Haar	10
2.2.	Filtros de ondícula para detección de bordes	11
2.3.	Base de Haar – Ejemplo de aplicación a una imagen	12
2.4.	Escalas en ondículas – Precisión y frecuencia	14
2.5.	Propagador de dispersión – Esquema del cálculo por capas	25
3.1.	Partición del plano de frecuencias con ondículas direccionales	46
3.2.	Red de dispersión – Arquitectura con ondículas y módulo	47
3.3.	Extractor convolucional – Arquitectura de red general	51
3.4.	Dígitos MNIST – Invariancia por traslaciones en capas profundas	60
3.5.	Dígitos MNIST – Modelado de variaciones de escritura	62
5.1.	Segmentación de instancias – Combinación de detección y segmentación	67
5.2.	Grad-CAM – Mapas de calor para clases gato y perro	68
5.3.	Resumen gráfico del enfoque propuesto en este trabajo	69
5.4.	Ciclo de vida en cascada con retroalimentación	71
5.5.	Planificación original del proyecto	72
5.6.	Planificación final del proyecto	72
6.1.	Tasa de aprendizaje – Importancia de una buena elección	75
6.2.	Perceptrón multicapa con una capa oculta	77
6.3.	Funciones de activación – Gráficas	78
6.4.	Retropropagación – Ejemplo en grafo computacional	79
6.5.	Red neuronal convolucional – Arquitectura y clasificación	81
6.6.	Red neuronal convolucional – Flujo de datos visual	82
6.7.	Mapa de activación – Cálculo en capa convolucional	83
6.8.	Capas convolucionales – Secuencia y profundidad de filtros	83
6.9.	Capa de <i>pooling</i> – Reducción espacial y <i>max-pooling</i>	84
6.10.	Capa totalmente conectada con dos capas ocultas	85
6.11.	Visión por Computador – Tareas aplicadas en una escena	86
6.12.	Clasificación de imágenes – Caballo y Persona	87
6.13.	Detección de objetos – Caballo y Persona	87
6.14.	Segmentación semántica – Caballo y Persona	88
6.15.	Segmentación de instancias – Caballo y Persona	89
6.16.	Arquitectura R-CNN	91
6.17.	Arquitectura Fast R-CNN	92
6.18.	Arquitectura Faster R-CNN	93

Índice de figuras

6.19. Exactitud frente a interpretabilidad para distintos algoritmos de Aprendizaje Automático	94
7.1. Publicaciones en Scopus – Segmentación dental	100
7.2. Publicaciones en Scopus – Segmentación dental con Mask R-CNN	101
7.3. Publicaciones en Scopus – Técnicas de explicabilidad y Grad-CAM en Mask R-CNN	103
8.1. Evolución de R-CNN hasta Mask R-CNN	108
8.2. Arquitectura Mask R-CNN	109
8.3. <i>Backbone</i> con FPN – Rutas y conexiones de salto	110
8.4. Ejemplo de interpolación por vecino más cercano	110
8.5. <i>Anchor boxes</i> – Escalas y relaciones de aspecto	112
8.6. Comparación entre <i>RoI Pooling</i> y <i>RoI Align</i>	113
8.7. Mask R-CNN – Resultados en el conjunto COCO	114
8.8. Grad-CAM – Localización según profundidad de capa	117
8.9. Adaptación de Grad-CAM a Mask R-CNN	120
9.1. Proyecto Mask R-CNN – Diagrama de paquetes	123
9.2. Proyecto Mask R-CNN – Diagrama de clases del paquete model	124
9.3. Proyecto Mask R-CNN – Diagrama de clases del paquete config	125
9.4. Proyecto Mask R-CNN – Diagrama de clases del paquete parallel_model	126
9.5. Proyecto Mask R-CNN – Diagrama de clases del paquete utils	126
9.6. Diagrama de secuencia	126
10.1. Notación FDI – Registro dental en odontología	128
10.2. Ejemplos por categoría del conjunto de datos DNS Panoramic Images	130
10.3. Máscara dental – Segmentación por categoría	131
10.4. Ejemplos de radiografías con segmentación dental	132
10.5. Distribución de métricas por partición – Modelo base	140
10.6. Distribución de métricas por partición – Modelo ajustado	141
10.7. Distribución de métricas de explicabilidad – Modelo base	144
10.8. Distribución de métricas de explicabilidad – DentCAM-RCNN	144
10.9. Comparación completa por categoría (1 a 4)	145
10.10. Comparación completa por categoría (7 a 10)	146
10.11. DentCAM-RCNN frente a métodos clásicos de segmentación no supervisados – Categorías 1 a 4	149
10.12. DentCAM-RCNN frente a métodos clásicos de segmentación no supervisados – Categorías 7 a 10	150
B.1. Evolución de las funciones de pérdida – Modelo ajustado	159

Índice de tablas

7.1. Estudios con Mask R-CNN y Grad-CAM para segmentación dental	105
10.1. Características del conjunto de datos DNS Panoramic Images	129
10.2. Distribución del conjunto de datos DNS Panoramic Images por subconjuntos y categorías	131
10.3. Métricas promedio de detección y segmentación – Modelos base y ajustado .	139
10.4. Métricas promedio de explicabilidad – Modelos base y ajustado	143
10.5. Comparación de métricas de evaluación globales – Métodos no supervisados frente a DentCAM-RCNN	148
10.6. Comparativa de DentCAM-RCNN con estudios recientes de segmentación de instancias dentales en 2D	151
A.1. Métricas de detección y segmentación por categoría – Modelo base	156
A.2. Métricas de explicabilidad por categoría – Modelo base	157
B.1. Métricas de detección y segmentación por categoría – Modelo ajustado	160
B.2. Métricas de explicabilidad por categoría – Modelo ajustado	161

Parte I.

Análisis de Redes Convolucionales

1. Introducción

Las redes neuronales convolucionales (véase [Sección 6.2](#)) se han establecido como una herramienta preeminente en el ámbito de la Inteligencia Artificial (IA), específicamente en el campo del Aprendizaje Profundo. Esta rama del Aprendizaje Automático (véase [Sección 6.1](#)) 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. Aunque la mayoría de los avances han sido de carácter empírico, también se han desarrollado análisis teóricos relevantes [[JS08](#), [BPL10](#)]. 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 (véase [Sección 6.3](#)), 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.

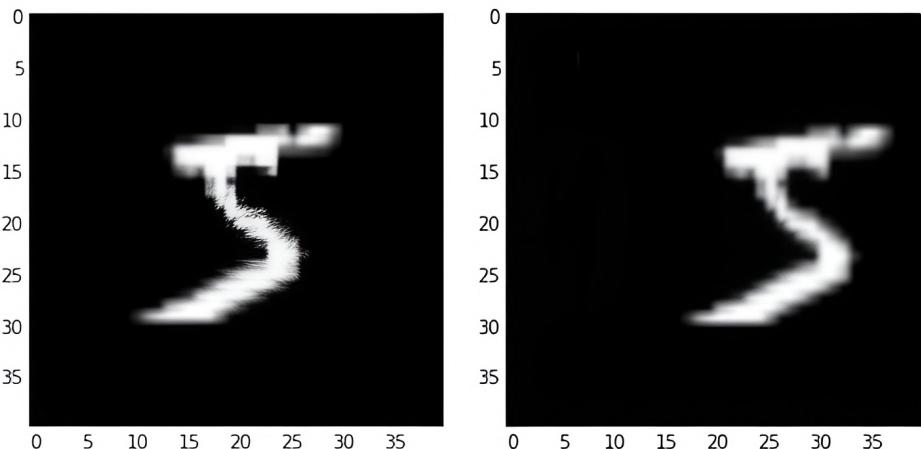


Figura 1.1.: Dígitos manuscritos del conjunto de datos MNIST [[LC98](#)]. Ejemplo de invarianza por traslaciones. A pesar de la variación en la ubicación del dígito manuscrito 5 en las imágenes de la izquierda y la derecha, este se identifica de manera consistente. Imagen extraída de [[KA18](#)].

1. Introducción

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 $L^2(\mathbb{R}^d) := \left\{ f : \mathbb{R}^d \rightarrow \mathbb{C} \mid \int_{\mathbb{R}^d} |f(x)|^2 dx < \infty \right\}$. 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).

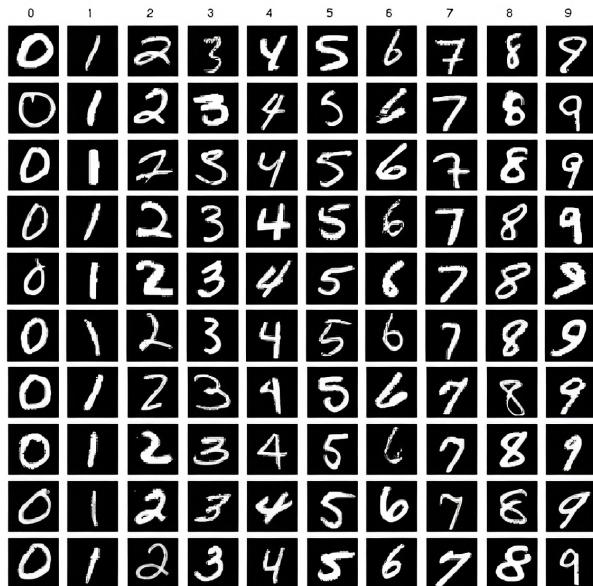


Figura 1.2.: Dígitos manuscritos del conjunto de datos MNIST [LC98]. Ejemplo de invarianza frente a pequeñas deformaciones. A pesar de las variaciones en la forma de los dígitos manuscritos, estos se identifican correctamente. Imagen extraída de [LYP16].

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 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.

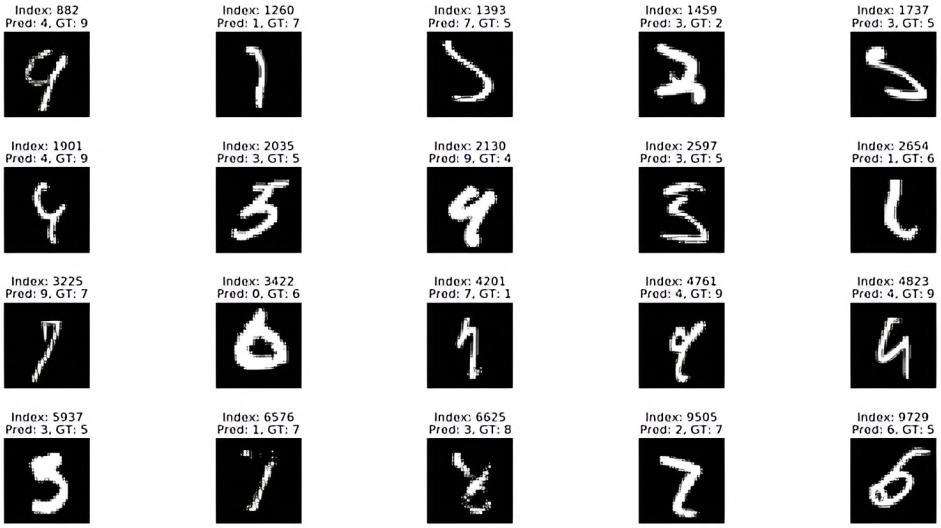


Figura 1.3.: Dígitos manuscritos del conjunto de datos MNIST [LC98]. Ejemplos de fallos de predicción causados por deformaciones en los dígitos. Las deformaciones pueden llevar a confusiones entre diferentes clases, como se observa en las imágenes donde la predicción (Pred) y la etiqueta real (GT) no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

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:

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

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 $\|f\| d(\mathbb{1}, \mathbb{1} - \tau)$

1. Introducción

τ). 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 $\|\tau\|_\infty := \sup_{x \in \Omega} |\tau(x)|$. 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 commutativos, 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 traslaciones, en el sentido de que las características se vuelven progresivamente más invariantes a las traslaciones 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, 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:

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

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 $0 < \epsilon \ll 1$, $\|(Jac(\tau))(x)\|_\infty = \epsilon$ y $\|(Hess(\tau))(x)\|_\infty = 0$. 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:

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

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.

2.1.2. 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

2. Teoría de Mallat sobre Redes de Dispersión

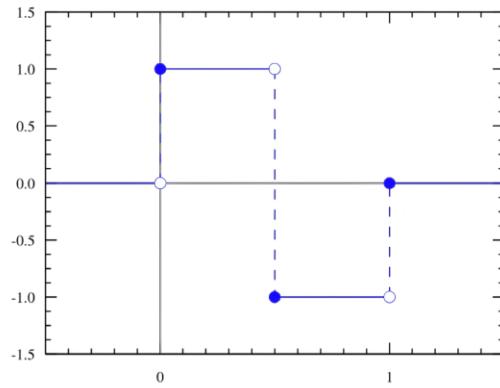
características como transiciones y singularidades. En imágenes, los coeficientes más grandes de las ondículas se concentran alrededor de zonas con esquinas o texturas irregulares.

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](#).



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

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:

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

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 $L^2(\mathbb{R})$:

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

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.

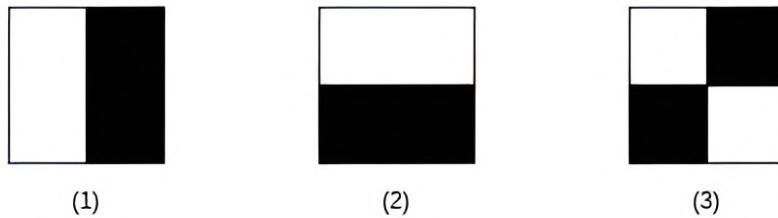


Figura 2.2.: Filtro generado por el soporte de una ondícula para detectar bordes (1) verticales, (2) horizontales y (3) diagonales.

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).

2. Teoría de Mallat sobre Redes de Dispersión

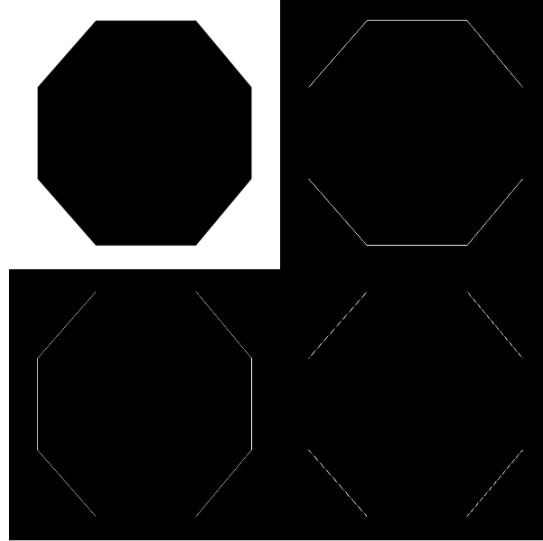


Figura 2.3.: Ejemplo de aplicación de la base de Haar a una imagen (arriba izquierda). Los filtros destacan los bordes en tres direcciones: horizontal (arriba derecha), vertical (abajo izquierda) y diagonal (abajo derecha). Imagen extraída de [Jor24].

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:

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

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 denotar 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](#)):

2. Teoría de Mallat sobre Redes de Dispersión

- 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:

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

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:

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

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:

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

2. Teoría de Mallat sobre Redes de Dispersión

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

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 $\beta = \frac{1}{2}$. \square

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:

- 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 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. \square

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:

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

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.

2.2.2. El operador módulo

En nuestro caso, para garantizar la Lipschitz-continuidad bajo difeomorfismos, es necesario que $M[\lambda]$ commute 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 .

2. Teoría de Mallat sobre Redes de Dispersión

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. Un conjunto 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 conmutativos de la forma $U[\lambda]f = M[\lambda](W[\lambda]f) = |f * \psi_\lambda| = |\int_{\mathbb{R}^d} f(u)\psi_\lambda(x-u)du|$, 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 $\|\psi_\lambda\|_1 := \int |\psi_\lambda(x)|dx$ denota la norma de ψ_λ en $L^1(\mathbb{R}^d)$.

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

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

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:

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:

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

□

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

2. Teoría de Mallat sobre Redes 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 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 $S_J f$ 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

2. Teoría de Mallat sobre Redes de Dispersión

anteriormente como:

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

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

Al calcular $U_J f$, 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 $A_J U[p] = S_J[p]$, lo que 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](#)).

2.3.2. Comparativa con una red neuronal convolucional

Las operaciones de la transformada de dispersión de ventanas 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 retropropagación 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,

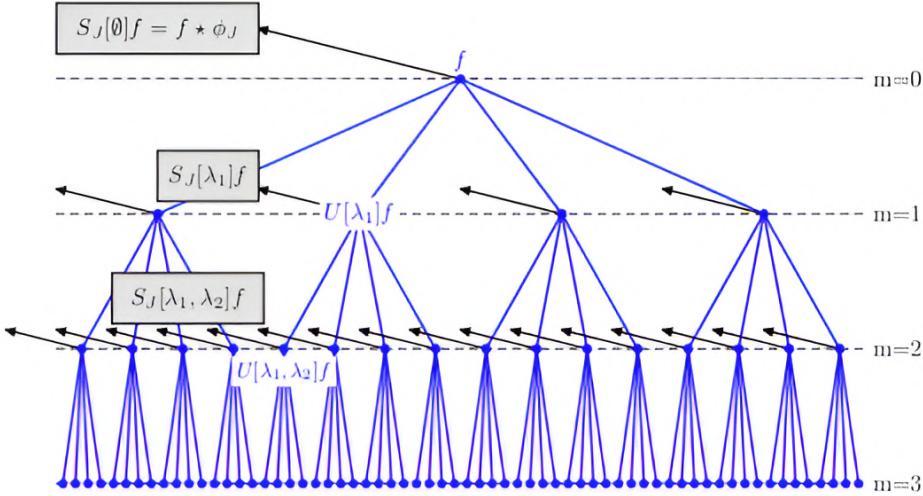


Figura 2.5.: Un propagador de dispersión U_J aplicado a un punto de una señal $f(x)$ calcula $(U[\lambda_1]f)(x) = |(f * \psi_{\lambda_1})(x)|$. En la capa $m = 0$, se promedian los coeficientes que dieron 0 (por tener $2^j < 2^{-J}$), obteniendo como salida $(S_J[\emptyset]f)(x) = (f * \phi_{2J})(x)$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

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. Teoría de Mallat sobre Redes de Dispersión

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:

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

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. \square

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 $\hat{\psi}_\lambda$, 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

2.3. Propagación de la dispersión y conservación de la norma

se aproxima a 0 al aumentar la longitud del camino, el teorema que demostraremos 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. \square

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 $|\hat{\rho}(\omega)| \leq |\hat{\phi}(2\omega)|$ y $\hat{\rho}(0) = 1$, tal que la función

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

satisface:

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

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:

2. Teoría de Mallat sobre Redes de Dispersión

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

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:

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

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:

2.3. Propagación de la dispersión y conservación de la norma

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

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:

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

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. \square

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:*

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

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:

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

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:

2. Teoría de Mallat sobre Redes de Dispersión

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

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:

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

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:

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

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 $\lim_{J \rightarrow \infty} \|S_J[\mathcal{P}_J]f\| = \|f\|$.

2. Teoría de Mallat sobre Redes de Dispersión

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 $\|Jac(\tau)\|_\infty \leq \frac{1}{2}$ se cumple que:*

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

Demostración. La norma 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:

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

Si ahora calculamos el módulo obtenemos que:

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

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

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

Por otro lado, vamos a comprobar que:

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

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

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

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

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

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

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

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

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

2. Teoría de Mallat sobre Redes de Dispersión

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:

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

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 translaciones.

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

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

Demostración. Fijamos $f \in L^2(\mathbb{R}^d)$. Teniendo en cuenta la comutatividad $S_J[\mathcal{P}_J](T_t f) = T_t(S_J[\mathcal{P}_J]f)$ 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 $\|U[\mathcal{P}_J]f\|$:

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

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 comutadores 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 comutador 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 $L \in L^2(\mathbb{R}^d)$ y cualquier $f \in L^2(\mathbb{R}^d)$, se cumple que:*

2. Teoría de Mallat sobre Redes de Dispersión

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

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:

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

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, demostrarímos 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:

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

Con m sustituciones, obtenemos:

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

Vemos así que $[S_J[\mathcal{P}_{J,m}], L] = \sum_{n=0}^m K_{m-n} V_J^n$. Finalmente veamos que $\|K_m\| \leq \|U_J\|$, 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:

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

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:

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

2. Teoría de Mallat sobre Redes de Dispersión

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 conmutador 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:

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

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 multivel 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:

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

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

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

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

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:

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

dónde $\lambda = (j, k) \in \Lambda_W \setminus \{(-J,0)\}$ representa ondículas direccionales [Malo8], [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:

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

dónde $\langle f, g \rangle := \int_{\mathbb{R}^d} f(x) \overline{g(x)} dx$, 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.

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.

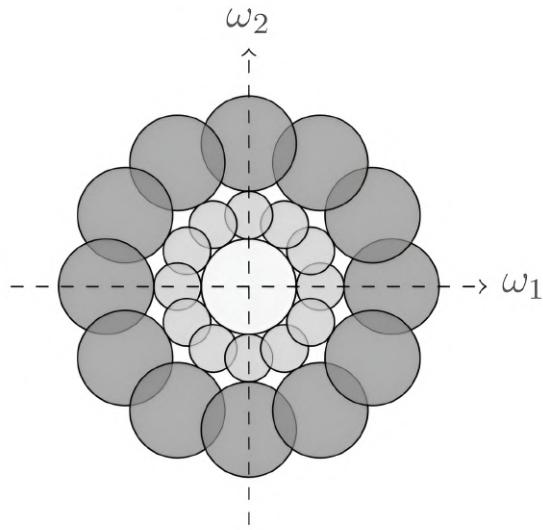


Figura 3.1.: Partición del plano de frecuencias \mathbb{R}^2 inducida por un marco semi-discreto de ondículas direccional, con $K = 12$ direcciones. Imagen extraída de [WB18].

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 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], [RPCL06].

3.2. Construcción del extractor de características convolucional general

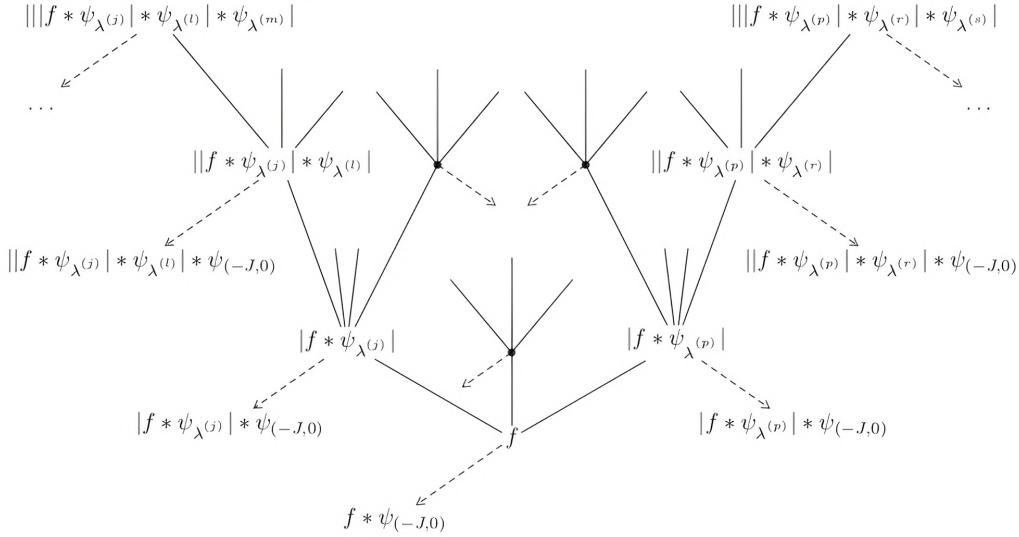


Figura 3.2.: Arquitectura de red de dispersión basada en filtros de ondículas y la no-linealidad del módulo. Los elementos del vector de características $\Phi_W(f)$ se indican en las puntas de las flechas. Imagen extraída de [WB18].

- 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 [PCDo8], *pooling* promedio [HLo6], [JKRL09], o *max-pooling* [SWPo5], [MLo6].
- 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:

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

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

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 [HL06], [JKRL09]. 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 $Idf = 0$ para $f = 0$).

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

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

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 sucesión de ternas:

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

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

se denomina *sucesión 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 sucesión 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 $\Lambda_1^n := \Lambda_1 \times \Lambda_2 \times \cdots \times \Lambda_n$. Un conjunto ordenado $q = (\lambda_1, \lambda_2, \dots, \lambda_n) \in \Lambda_1^n$ 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:

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

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:

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

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

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

3.2. Construcción del extractor de características convolucional general

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 sucesión 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.

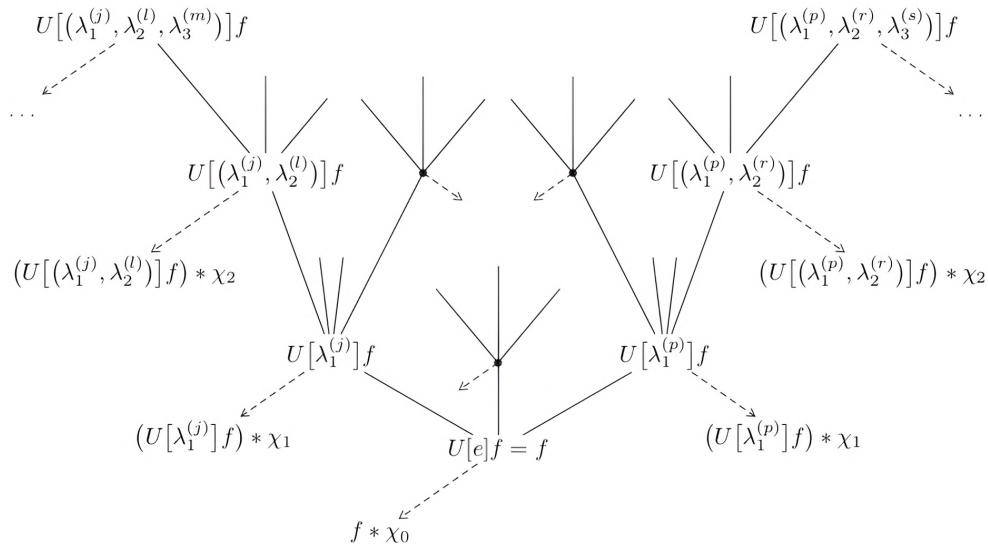


Figura 3.3.: Arquitectura de red subyacente al extractor de características convolucional general. El índice $\lambda_n^{(k)}$ corresponde al k -ésimo átomo $g_{\lambda_n^{(k)}}$ del marco Ψ_n asociado a la capa n -ésima de la red. La función χ_n es el átomo generador de salida de la capa n -ésima. Imagen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

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

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

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

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

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 $\Phi_\Omega : L^2(\mathbb{R}^d) \rightarrow (L^2(\mathbb{R}^d))^\mathcal{Q}$, con $\mathcal{Q} := \bigcup_{n=0}^{\infty} \Lambda_1^n$, 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 sucesión de módulos Ω , formalizada a continuación.

Proposición 3.1. *Sea $\Omega = ((\Psi_n, M_n, P_n))_{n \in \mathbb{N}}$ una sucesión 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 de series telescopicas. Comenzamos escribiendo:

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

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

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

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:

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

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:

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

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:

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

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:

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

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:

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

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 sucesión 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:

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

se denomina *condición de admisibilidad para sucesiones de módulos*. Las sucesiones 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 [HLo6, SWPo5, 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 sucesión de módulos *admissible*, 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 $P_n : L^2(\mathbb{R}^d) \rightarrow L^2(\mathbb{R}^d)$ comutan con el operador de traslación T_t , es decir:

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

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:

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

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 decrecimiento:

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

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:

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

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

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 sucesiones 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:

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

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

para todo $n \in \mathbb{N}$, donde en (3.36) usamos (3.29), y en (3.37) aplicamos la la relación $\widehat{T_t f} =$

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

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

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

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:

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

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

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

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 decayamiento (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). \square

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 comuta con el operador de traslación T_t .

Notamos que la condición (3.30) se puede cumplir fácilmente tomando los átomos generadores 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, [HLo6], [SWP05], [MLo6], 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.

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

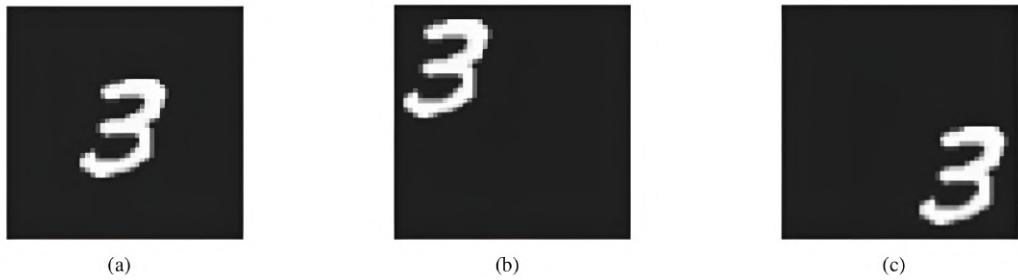


Figura 3.4.: Dígitos manuscritos del conjunto de datos MNIST [LC98]. Para tareas prácticas de Aprendizaje Automático (por ejemplo, clasificación de señales), a menudo deseamos que el vector de características $\Phi_\Omega(f)$ sea invariante a la ubicación espacial de los dígitos dentro de la imagen f . 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].

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 sucesión de módulos admisible, sea $S_n \geq 1$,*

3.3. Invarianza vertical por traslaciones

$n \in \mathbb{N}$, 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 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:

$$\|\Phi_\Omega^n(T_t f) - T_t(\Phi_\Omega^n(f))\|^2 = \|T_{t/(S_1 \cdots S_n)}\Phi_\Omega^n(f) - T_t(\Phi_\Omega^n(f))\|^2 \quad (3.47)$$

$$= \sum_{q \in \Lambda_1^n} \|(T_{t/(S_1 \cdots S_n)} - T_t)(f_q * \chi_n)\|^2$$

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

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:

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

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):

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

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:

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

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

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

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 $f \in L^2_R(\mathbb{R}^d) := \{f \in L^2(\mathbb{R}^d) \mid \text{supp}(\widehat{f}) \subseteq B_R(0)\}$, 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.

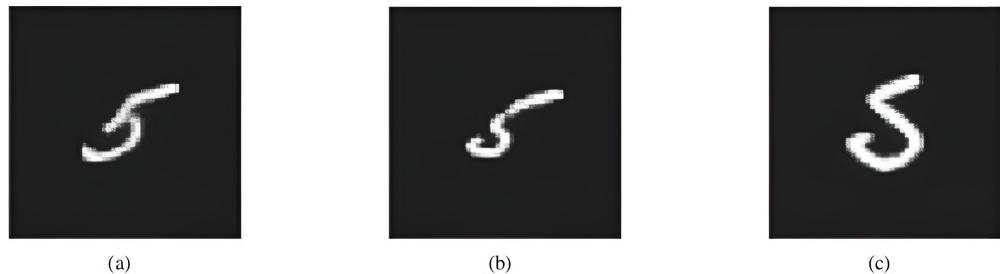


Figura 3.5.: Dígitos manuscritos del conjunto de datos MNIST [LC98]. Si f representa la imagen del dígito 5 en (a), entonces, con una función τ seleccionada apropiadamente, la función $T_\tau f = f(\cdot - \tau(\cdot))$ modela imágenes del 5 basadas en diferentes estilos de escritura como en (b) y (c). 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 transformada 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 [Sze22, TIF24], 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), dado que busca no solo identificar la presencia de un objeto en la imagen, sino también delimitar 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 se comportan 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.

En este trabajo se aplica la arquitectura Mask R-CNN a la segmentación de piezas dentales individuales en radiografías panorámicas, un caso de uso con gran relevancia práctica. Este tipo de imágenes es ampliamente utilizado en odontología y medicina forense, donde la identificación precisa de cada diente puede resultar clave, por ejemplo, en procesos de identificación humana cuando no se dispone de otros datos biométricos. La automatización de este análisis mediante modelos de aprendizaje profundo puede contribuir a mejorar la eficiencia, trazabilidad y objetividad en contextos periciales y clínicos, reduciendo la dependencia del análisis manual y potenciando el uso de herramientas de apoyo al diagnóstico.

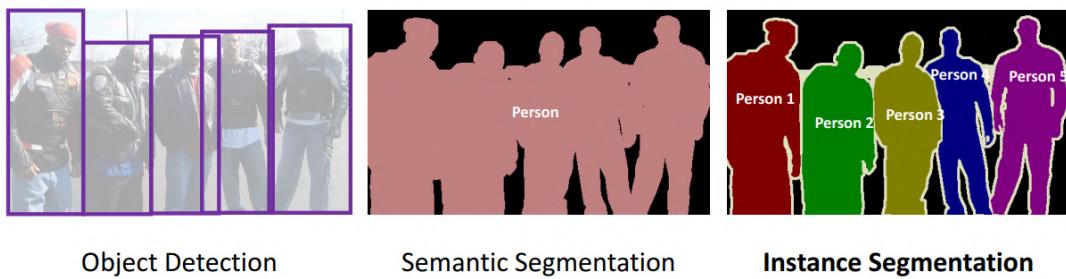


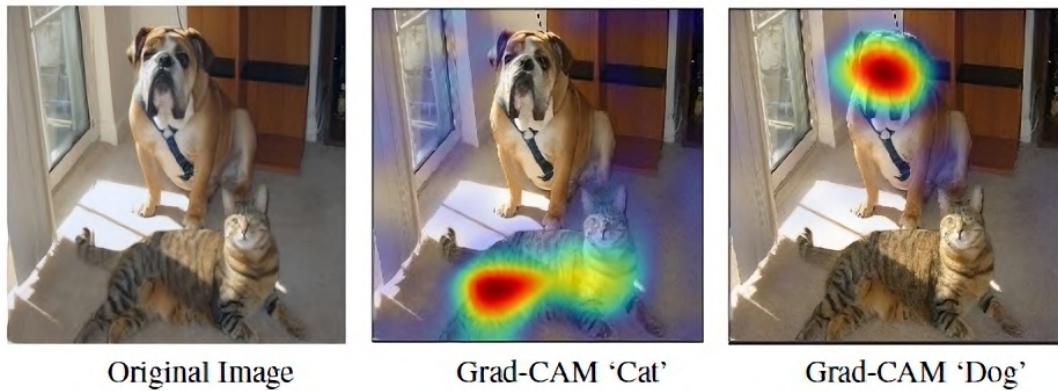
Figura 5.1.: La segmentación de instancias combina la detección de objetos y la segmentación semántica. Imagen extraída de [Liu18].

Este trabajo se centra en abordar el problema de la interpretabilidad de Mask R-CNN mediante la implementación de técnicas que aporten explicaciones visuales a sus decisiones. En particular, se aplica una adaptación de Grad-CAM [SCD⁺17] a dicha arquitectura, un método de interpretabilidad *post-hoc* que genera mapas de calor para visualizar las regiones

5. Introducción

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 (XAI, por sus siglas en inglés) [[Mol20](#), [LBC⁺24](#)].

Si bien Grad-CAM es ampliamente utilizado para explicar redes neuronales convolucionales, su adaptación efectiva a arquitecturas más complejas como Mask R-CNN plantea desafíos específicos, dado que esta última combina componentes distintos para detección y segmentación. Por ello, implementar y evaluar una adaptación efectiva de Grad-CAM en Mask R-CNN contribuye significativamente al campo de la XAI para tareas de segmentación de instancias.



[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](#)].

Con el fin de clarificar visualmente el enfoque planteado en este trabajo, se ha elaborado la [Figura 5.3](#), que resume gráficamente las etapas clave del proceso desarrollado. En dicha figura se muestra cómo una imagen radiográfica dental es procesada por la arquitectura Mask R-CNN para obtener segmentaciones instanciadas de las piezas dentales, y cómo posteriormente se aplica la técnica Grad-CAM sobre las capas internas del modelo para generar mapas de calor interpretables. Esta visualización permite identificar las regiones de la imagen que han influido más en las decisiones del modelo, dotando de explicabilidad al proceso de segmentación y aportando mayor transparencia en el contexto del análisis médico. La figura pretende representar de forma esquemática el objetivo global del proyecto: combinar segmentación y explicabilidad para mejorar la comprensión del comportamiento del modelo.

Cabe señalar que, aunque la segmentación de instancias ya proporciona una salida visual interpretable en forma de máscaras, este trabajo propone una capa adicional de explicabilidad mediante Grad-CAM, dirigida a analizar la coherencia entre las regiones activadas

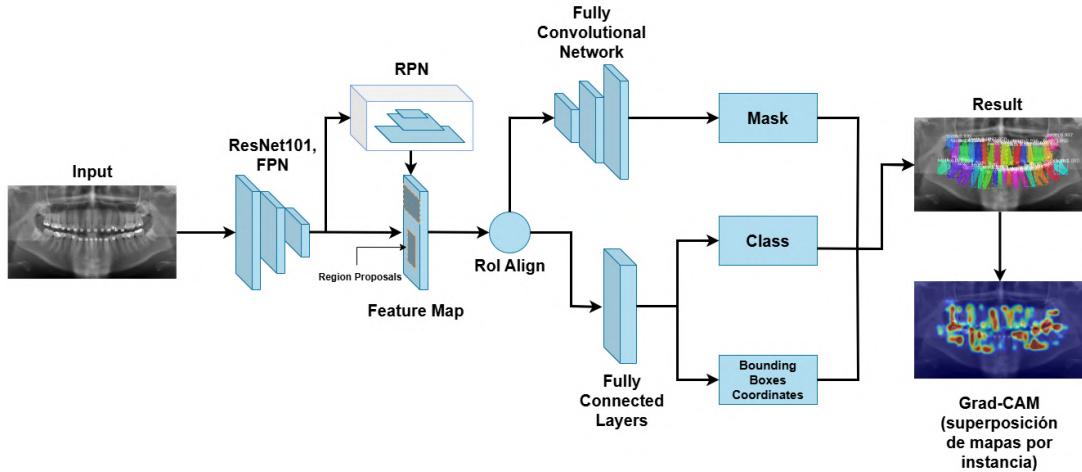


Figura 5.3.: Resumen visual del enfoque desarrollado en este trabajo. A partir de una imagen radiográfica dental de entrada, el modelo Mask R-CNN segmenta individualmente las piezas dentales y predice su clase. Una vez completada la inferencia, se aplica la técnica Grad-CAM para cada instancia detectada, generando mapas de atención específicos que se superponen finalmente sobre la imagen original. La caja etiquetada como *Grad-CAM (superposición de mapas por instancia)* representa esta visualización agregada. Esta explicación *post-hoc* permite interpretar visualmente las decisiones del modelo sin afectar al proceso de predicción.

internamente por la red y las regiones segmentadas externamente en radiografías dentales panorámicas. El objetivo no es explicar por qué se ha segmentado un píxel concreto, sino evaluar en qué medida la atención interna del modelo, estimada por los mapas de Grad-CAM, coincide con las estructuras dentales identificadas mediante segmentación. Esta consistencia entre ambos mapas puede ofrecer una señal indirecta sobre la confiabilidad y estabilidad del resultado: una activación dispersa o mal alineada con las máscaras podría indicar que el modelo se está apoyando en características espurias o poco relevantes; en cambio, una activación bien solapada con las regiones segmentadas sugiere un comportamiento coherente y robusto. En este sentido, la explicabilidad se orienta a validar la segmentación obtenida y aportar criterios visuales que permitan interpretar la confianza del modelo en el contexto específico del análisis automático de imágenes dentales.

5.2. Objetivos

El objetivo principal de este trabajo es **aplicar la arquitectura neuronal Mask R-CNN a la segmentación de radiografías dentales panorámicas y, posteriormente, dotarla de interpretabilidad mediante la adaptación e implementación de la técnica de explicabilidad Grad-CAM, para mejorar la comprensión de sus decisiones en tareas de segmentación de instancias dentales**. Para alcanzar este objetivo principal, se han definido los siguientes objetivos específicos:

5. Introducción

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. **Aplicar Mask R-CNN a radiografías dentales panorámicas:** Entrenar y ajustar la arquitectura Mask R-CNN sobre un conjunto de datos dentales con el fin de segmentar instancias dentales individuales en imágenes radiográficas, evaluando su rendimiento.
3. **Investigar el campo de la XAI:** 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.
4. **Implementar una adaptación de Grad-CAM en Mask R-CNN:** Aplicar y adaptar Grad-CAM en la arquitectura de Mask R-CNN con el objetivo de dotar de interpretabilidad a las predicciones realizadas por el modelo en el contexto de la segmentación de instancias dentales. Dicha implementación se basará en adaptaciones propuestas en investigaciones previas [XJ21, XVM⁺21, XVM⁺22].
5. **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, evaluando la calidad de las explicaciones generadas y determinando el grado en que facilitan la interpretabilidad de la segmentación de instancias dentales.

5.3. Planificación y estimación de costes

Desde el inicio de la planificación de este proyecto informático, se decidió emplear un enfoque basado en el modelo en cascada retroalimentado para el desarrollo del software [Pre05]. Esto se debe a que el proyecto consiste principalmente en la adaptación de un software ya existente, lo que llevó a estimar que esta fase no implicaría un alto nivel de complejidad. Tras dicha adaptación, se contempló la posibilidad de realizar pequeños ajustes al software según los resultados obtenidos durante la etapa de experimentación. Por este motivo, se optó por un modelo de ciclo de vida en cascada con retroalimentación, como se ilustra en [Figura 5.4](#).

Las fases llevadas a cabo en este proyecto son las siguientes:

1. **Análisis del problema y requisitos:** En esta etapa se ha examinado detalladamente el contexto del problema y su relevancia, además de realizar reuniones con los tutores del proyecto para definir claramente los requisitos y objetivos específicos del software a desarrollar.
2. **Diseño:** En primer lugar, se ha analizado la estructura del conjunto de datos que se ha utilizado. En segundo lugar, se ha llevado a cabo el diseño de los experimentos, incluyendo las métricas que se han empleado y los protocolos de validación que se han aplicado.
3. **Implementación:** En esta etapa se han desarrollado todos los experimentos diseñados en la fase previa.

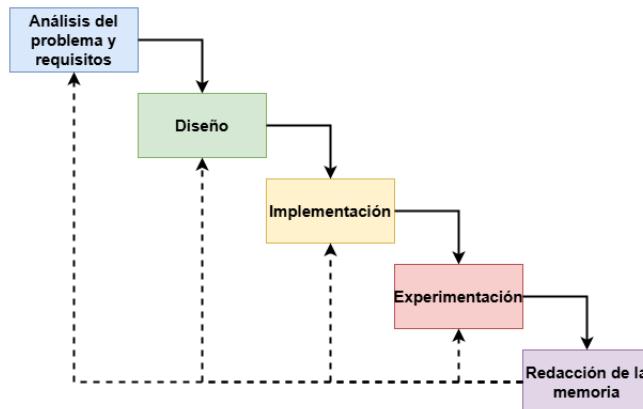


Figura 5.4.: Modelo de ciclo de vida en cascada con retroalimentación.

4. **Experimentación:** Se ha procedido a ejecutar los experimentos diseñados previamente y a analizar los resultados obtenidos. En base a algunos resultados, fue necesario revisar y ajustar algunas partes del software, volviendo a fases anteriores del ciclo de vida.
5. **Redacción de la memoria:** Finalmente, se ha documentado todo el trabajo realizado, incluyendo el desarrollo teórico, el diseño metodológico, los resultados experimentales y las conclusiones extraídas. Esta fase se desarrolló de forma progresiva durante las últimas etapas del proyecto y culminó en los meses finales con la elaboración completa del documento final.

Dado el elevado volumen de trabajo a lo largo del curso 2023-2024, dedicado principalmente a desarrollar la parte matemática de este trabajo, desde el inicio se organizó el desarrollo de este proyecto informático con la intención de presentar todo el trabajo en la convocatoria especial de noviembre del curso 2024-2025.

El desarrollo del proyecto se planificó considerando únicamente los días laborales de lunes a viernes. Se tenía pensado dedicar un mayor número de horas a las dos últimas fases en comparación con las dos primeras, debido a las restricciones de tiempo al compaginar este proyecto con una jornada laboral a tiempo completo desde julio de 2024. Durante este período, se estimó que no sería posible dedicar más de cuatro horas semanales al proyecto. La planificación original por meses del proyecto se puede observar en la [Figura 5.5](#).

Bajo este esquema inicial, las horas dedicadas fueron las siguientes:

- Primera fase (análisis del problema y requisitos): 75 horas.
- Segunda fase (diseño): 65 horas.
- Tercera fase (implementación): 110 horas.
- Cuarta fase (experimentación): 100 horas.
- Quinta fase (redacción de la memoria): 60 horas.

5. Introducción

Fase	Duración (en semanas)	Junio (2024)	Julio (2024)	Agosto (2024)	Septiembre (2024)	Octubre (2024)	Noviembre (2024)
Ánalisis del problema y requisitos	8						
Diseño	8						
Implementación	12						
Experimentación	12						
Redacción de la memoria	8						

Figura 5.5.: Planificación original del proyecto.

En total, se estimó un tiempo de desarrollo del proyecto de 410 horas. Sin embargo, esta planificación resultó ser insuficiente para completar el trabajo, lo que provocó un retraso en la entrega hasta junio del curso 2024-2025. La planificación final por meses del proyecto se puede observar en la [Figura 5.6](#). Las horas dedicadas al proyecto se incrementaron a:

- Primera fase (análisis del problema y requisitos): 90 horas.
- Segunda fase (diseño): 80 horas.
- Tercera fase (implementación): 140 horas.
- Cuarta fase (experimentación): 160 horas.
- Quinta fase (redacción de la memoria): 100 horas.

Fase	Duración (en semanas)	Junio (2024)	Julio (2024)	Agosto (2024)	Septiembre (2024)	Octubre (2024)	Noviembre (2024)	Diciembre (2024)	Enero (2025)	Febrero (2025)	Marzo (2025)	Abril (2025)	Mayo (2025)
Ánalisis del problema y requisitos	12												
Diseño	12												
Implementación	12												
Experimentación	16												
Redacción de la memoria	20												

Figura 5.6.: Planificación final del proyecto.

En total, el tiempo invertido ascendió a 570 horas. Inicialmente, el proyecto tenía un presupuesto de 10.250 €, considerando una tarifa de 25 €/hora para un investigador en una empresa tecnológica [\[Gla25\]](#). Sin embargo, el aumento en las horas trabajadas incrementó el presupuesto a 14.250 €. Además del coste en horas de trabajo, se ha considerado el uso de recursos materiales durante el desarrollo del proyecto. En particular, los experimentos se ejecutaron en un portátil ASUS TUF Gaming F17 FX707VI-HX040, cuyo precio de adquisición es de aproximadamente 1.500 €. Teniendo en cuenta una vida útil estimada del equipo de cuatro años y un uso parcial de alrededor del 80 % durante los doce meses en los que se desarrolló este trabajo, se ha aplicado una amortización proporcional. De este modo, se estima un coste asociado al uso del hardware de unos 300 €, lo que eleva el presupuesto total del proyecto a 14.550 €.

6. Fundamentos Teóricos

En este capítulo se presentan los conceptos teóricos esenciales que sirven de base para comprender y desarrollar este trabajo. En particular, se abordarán con detalle temas fundamentales como el Aprendizaje Automático, las redes neuronales y técnicas básicas en Visión por Computador, ya que estos conceptos sustentan la arquitectura Mask R-CNN empleada posteriormente. Además, se introducirá la XAI, ya que uno de los objetivos centrales de este trabajo consiste en dotar de interpretabilidad a dicha arquitectura mediante técnicas como Grad-CAM. Para ello, se ha utilizado conocimiento adquirido en asignaturas como Inteligencia Artificial, Visión por Computador y Aprendizaje Automático, además de artículos especializados mencionados a lo largo del capítulo. Asimismo, parte de la información expuesta proviene del curso CS231n de la Universidad de Stanford [Unizo].

6.1. Aprendizaje Automático

La IA ha estado presente en el desarrollo de soluciones informáticas desde hace décadas, con aplicaciones tempranas que abarcan desde algoritmos de búsqueda como Dijkstra o A*, hasta sistemas de lógica difusa en electrodomésticos o programas capaces de jugar a las damas desde los años cincuenta. Su objetivo principal es dotar a los ordenadores de la capacidad de razonar y resolver problemas de forma autónoma e inteligente [RN22]. A lo largo del tiempo, se han desarrollado múltiples estrategias dentro de la IA para alcanzar este objetivo, entre las que destacan las Metaheurísticas y, de manera especial, el Aprendizaje Automático (*Machine Learning* en inglés) [Mur22, Mur23, Bis06, AMMIL12]. Dado que este trabajo se centra en dicha área, resulta fundamental introducir y definir con claridad este concepto.

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.

6. Fundamentos Teóricos

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 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, el objetivo es encontrar una función $f(x) = y$ que relacione una entrada $x \in \mathbb{R}^m$ con una salida continua $y \in \mathbb{R}^n$. Como esta función suele ser desconocida, se aproxima mediante un modelo parametrizado $\hat{f}(x, w)$, donde w representa los parámetros ajustables del modelo.

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 utilizados para entrenar el modelo.
- P = Función de coste que mide el error de predicción.

Un ejemplo común es el modelo lineal, donde la predicción se realiza mediante:

$$\hat{f}(x, w) = w^T x,$$

y el error se mide utilizando el error cuadrático medio:

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

El aprendizaje consiste en encontrar el vector w que minimiza esta función de coste \mathcal{L} , ajustando el modelo a los datos de entrenamiento.

6.1.1.2. Problemas de clasificación

En los problemas de clasificación, el objetivo es asignar cada entrada a una clase específica. Un caso habitual es la clasificación binaria, donde se predice si un dato pertenece a una clase (0 o 1). También existen problemas de clasificación multiclas, donde hay más de dos categorías posibles.

Para resolver estas tareas, se buscan fronteras que separan con precisión los diferentes grupos de datos. Algunos algoritmos comúnmente empleados son:

- **K-nearest neighbours:** Clasifica según las etiquetas de los K vecinos más cercanos [Koz08].

- **Máquinas de vectores de soporte (SVM):** Encuentran el hiperplano que maximiza la separación entre clases [HDO⁺98].
- **Redes neuronales:** Permiten aprender límites no lineales complejos (profundizaremos en ellas más adelante) [Bis94].

6.1.1.3. Gradiente descendente

Una vez planteado un problema de Aprendizaje Automático, es necesario optimizar los parámetros del modelo para minimizar una función de coste derivable. Para ello, uno de los algoritmos más utilizados es el *gradiente descendente* [Rud16].

El gradiente de la función de coste indica la dirección de máximo crecimiento. Por tanto, actualizando los parámetros en la dirección opuesta, podemos acercarnos a un mínimo:

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

donde η es la tasa de aprendizaje, que controla el tamaño del paso en cada iteración.

Es importante elegir adecuadamente η : si es demasiado pequeña, el entrenamiento será lento; si es demasiado grande, el algoritmo puede no converger (véase Figura 6.1). A menudo se utilizan estrategias de ajuste dinámico de la tasa de aprendizaje para mejorar la estabilidad y eficiencia del entrenamiento.

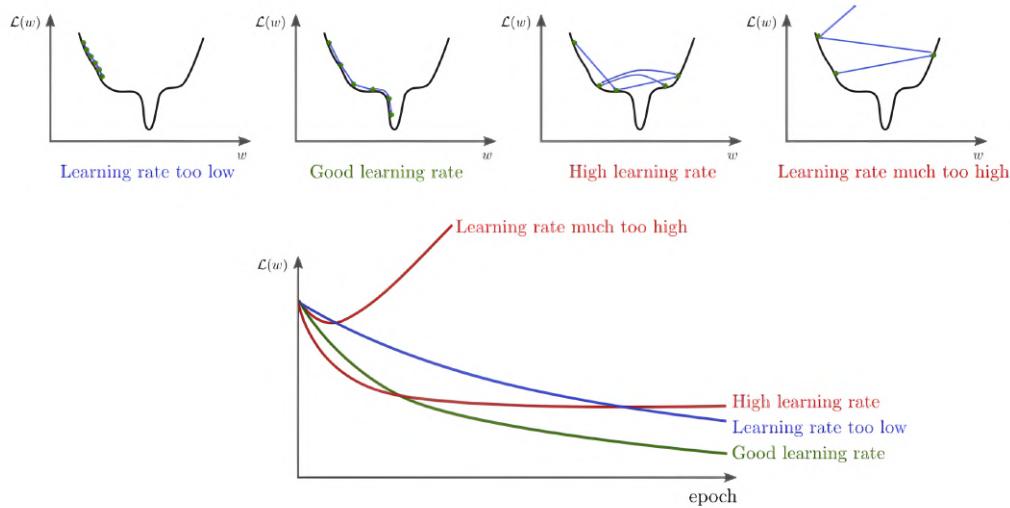


Figura 6.1.: Importancia de una elección adecuada de la tasa de aprendizaje. Imagen extraída de [Uni20].

6. Fundamentos Teóricos

6.1.2. Aprendizaje No Supervisado

Aunque este tipo de aprendizaje no se ha utilizado directamente en el presente trabajo, se incluye brevemente por su relevancia general dentro del campo del Aprendizaje Automático. 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.

Aunque este tipo de aprendizaje no se ha utilizado directamente en el presente trabajo, se incluye brevemente por su relevancia general dentro del campo del aprendizaje automático. 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.

6.1.3. 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, Pri23, BB23] 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.

6.1.4. 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):

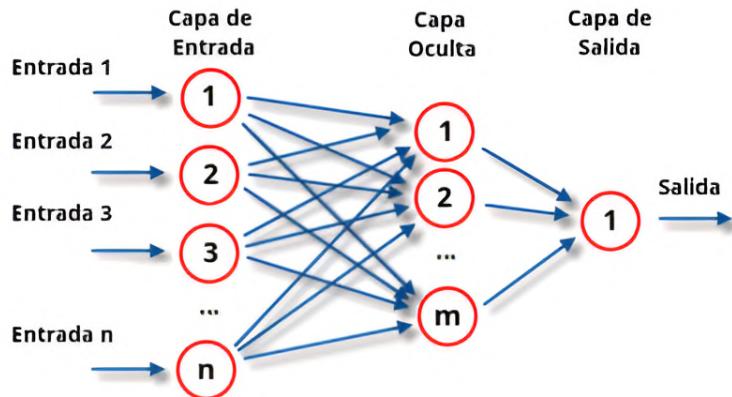


Figura 6.2.: Perceptrón multicapa con una capa oculta. Formalmente, se puede expresar como $\hat{f}(x, w) = f_2(f_1(x, w))$, donde f_1 representa la transformación de los datos de la capa de entrada a la capa oculta, y f_2 es la transformación de la capa oculta a la capa de salida. Imagen extraída de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

- **Función 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 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 retropropagación.

- **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

6. Fundamentos Teóricos

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 multiclas.

$$\text{softmax}(x)_j = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}}, \quad j = 1, \dots, K.$$

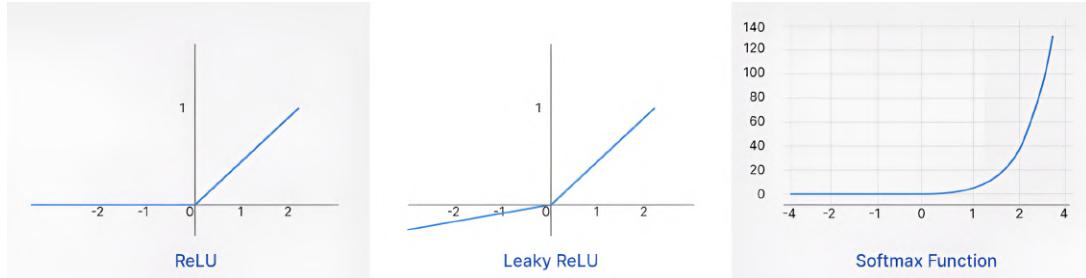


Figura 6.3.: Gráficas de las funciones de activación más utilizadas. Imagen extraída de [\[Onl24\]](#).

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:

$$f_i(x, w) = \gamma(w_i^T x),$$

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.

6.1.4.1. Retropropagación

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 retropropagación (*backpropagation* en inglés).

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 retropropagación 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, se procede a actualizar los pesos, utilizando métodos como el algoritmo de gradiente descendente.

Para comprender mejor este procedimiento, véase Figura 6.4.

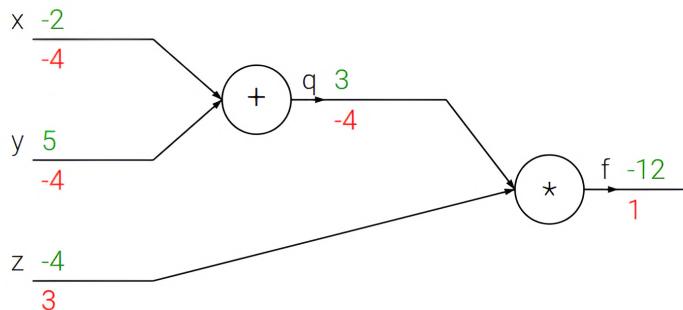


Figura 6.4.: Ejemplo de retropropagación representando la función $f(x, y, z) = (x + y)z$, para valores $x = -2$, $y = 5$, $z = -4$, en un grafo computacional. El grafo muestra el flujo de cálculo de la salida $f = -12$. Primero se calculan los valores desde las entradas hasta la salida (mostrados en verde). Luego, se realiza la retropropagación, 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].

6.2. 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

6. Fundamentos Teóricos

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](#)).

6.2.1. 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*.

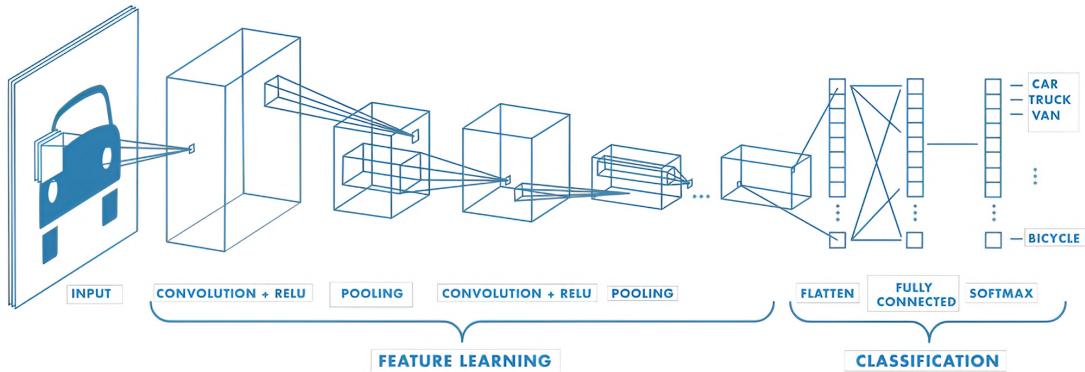


Figura 6.5.: Diagrama esquemático de la arquitectura de una red neuronal convolucional, que ilustra el proceso de aprendizaje de características y la etapa de clasificación. Las capas de convolución y *pooling* permiten la extracción progresiva de características significativas, mientras que las capas finales totalmente conectadas realizan la clasificación en categorías predefinidas, como coche o bicicleta. Imagen extraída de [Mat24b].

- **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 emplean 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. Fundamentos Teóricos

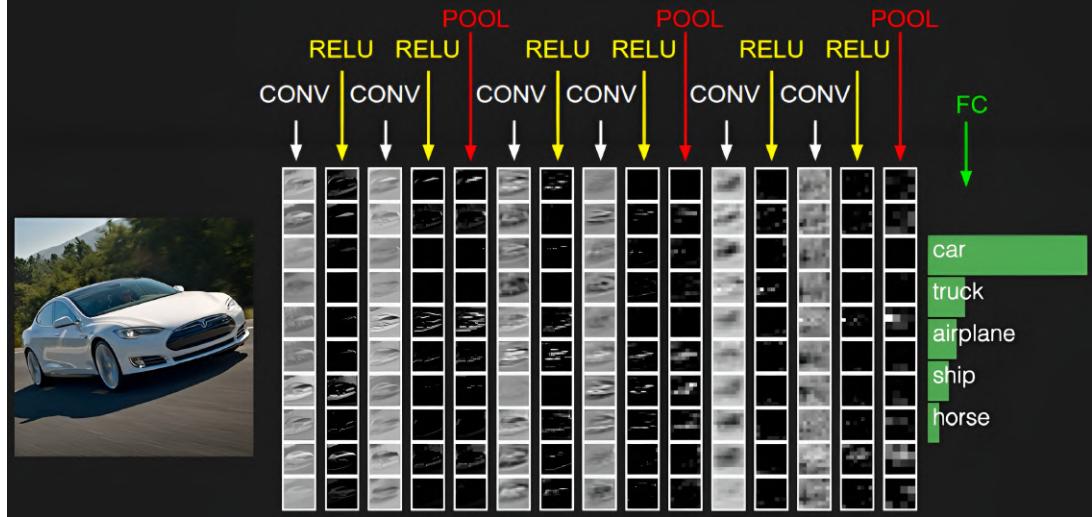


Figura 6.6.: Ejemplo visual del flujo de datos en una red neuronal convolucional, donde se observa cómo las capas de convolución (CONV) y activación (ReLU) extraen características de la imagen de entrada, seguidas de capas de *pooling* (POOL) que reducen la dimensión espacial manteniendo la profundidad. Al final, una capa totalmente conectada (FC) clasifica la imagen en una categoría específica, siendo coche en este caso. Imagen extraída de [Unizo].

6.2.2. 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 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.

6.2.3. 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

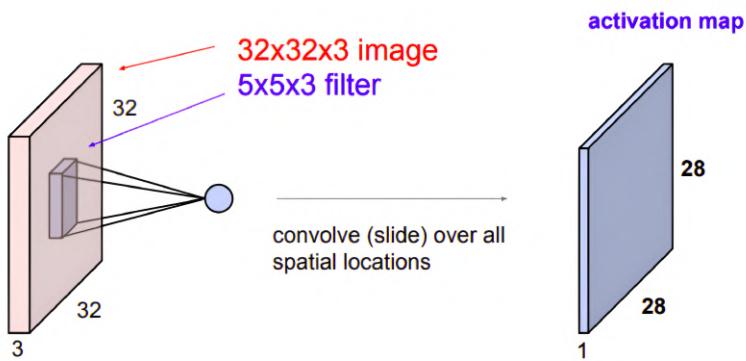


Figura 6.7.: Ejemplo que ilustra cómo se calcula un mapa de activación en una capa convolucional para un volumen de entrada específico. Imagen extraída de [Uni2o].

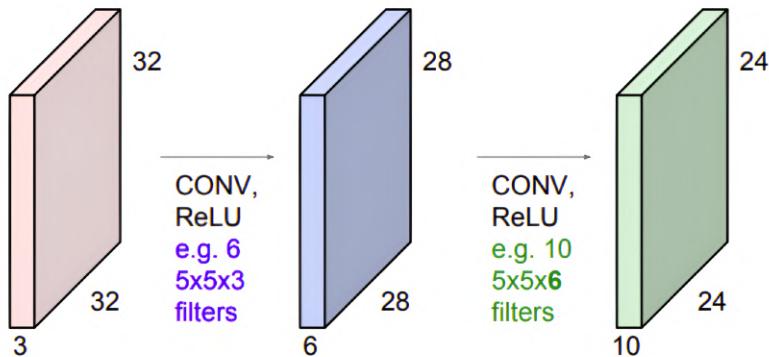


Figura 6.8.: Secuencia de varias capas convolucionales seguidas de la función de activación. Es importante señalar que la profundidad de los filtros siempre coincide con la del volumen de entrada. Imagen extraída de [Uni2o].

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.

6.2.4. 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

6. Fundamentos Teóricos

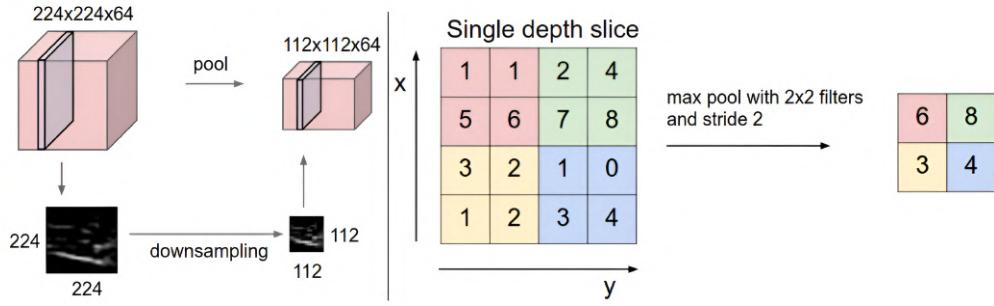


Figura 6.9.: La capa de *pooling* reduce espacialmente el tamaño del volumen de entrada de forma independiente en cada capa de profundidad. Izquierdo: En este ejemplo, el volumen de entrada de tamaño $224 \times 224 \times 64$ se reduce mediante un filtro de tamaño 2 y un stride de 2, obteniendo un volumen 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 [Uni2o].

tes 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.

6.2.5. 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 retropropagación.
- 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.).

6.2.6. Fine-tuning

El *fine-tuning* es una técnica habitual en Aprendizaje Profundo para adaptar modelos pre-entrenados a nuevas tareas o conjuntos de datos específicos [PY10]. En lugar de entrenar desde cero, se ajustan algunos parámetros del modelo previamente entrenado, lo que mejora

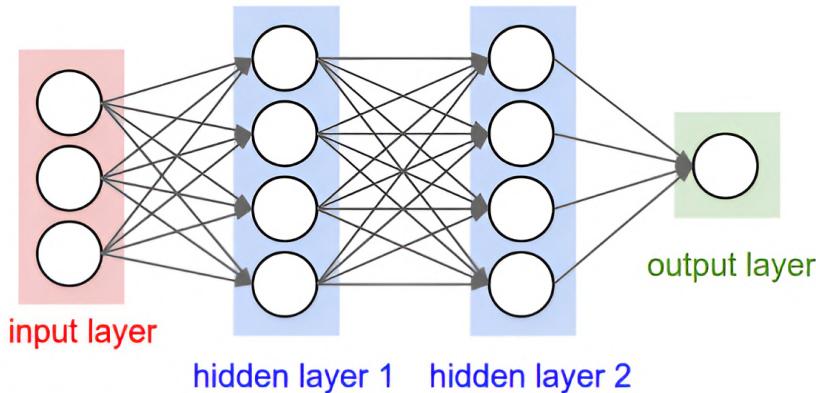


Figura 6.10.: Ejemplo de capa totalmente conectada con dos capas ocultas. Imagen extraída de [Uni20].

la eficiencia y precisión [YCBL14].

El proceso consiste en descongelar una o más capas superiores del modelo, manteniendo fijas las capas inferiores, que conservan características generales aprendidas, como bordes o texturas [Ben12]. Esto permite aprovechar el conocimiento previo y reducir tanto el tiempo de entrenamiento como la necesidad de datos.

En Visión por Computador, se emplea frecuentemente para adaptar modelos entrenados en conjuntos como ImageNet a tareas específicas, como detección de objetos o clasificación médica [HAE16]. Ha demostrado mejorar la precisión y reducir los costes computacionales frente al entrenamiento desde cero [KSL19].

6.3. Visión por Computador

La Visión por Computador (*Computer Vision* en inglés) es una disciplina que combina varias áreas del conocimiento, como la IA 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.

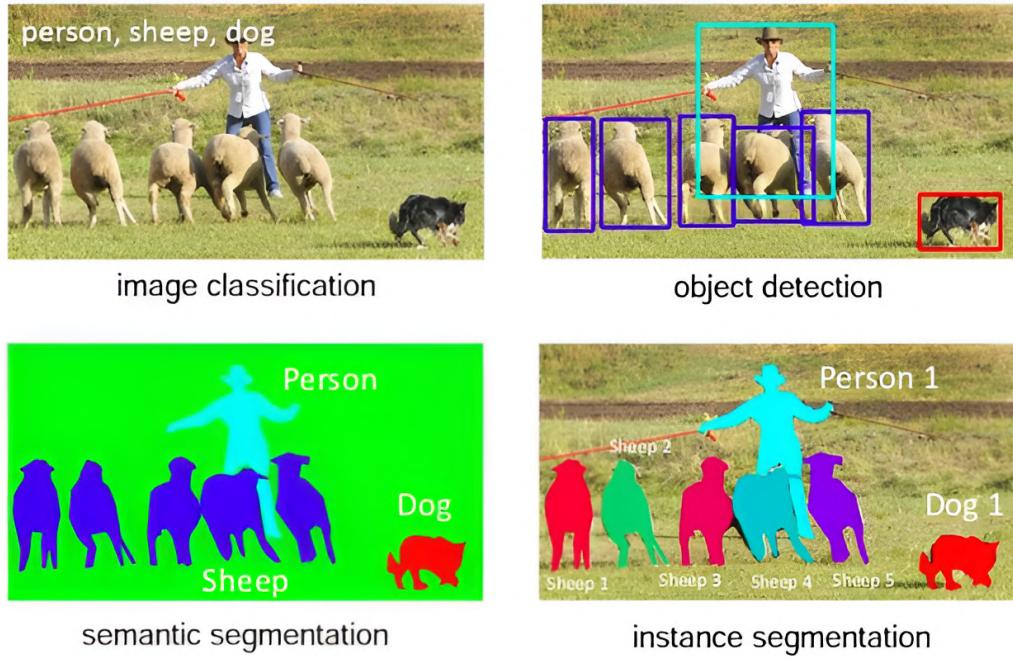


Figura 6.11.: Diferentes tareas de Visión por Computador: Clasificación de imágenes, detección de objetos, segmentación semántica y segmentación de instancias aplicadas en una misma escena. Imagen extraída de [LOW⁺19].

6.3.1. Clasificación de imágenes

El objetivo de esta tarea en Visión por Computador es identificar las categorías presentes en una imagen (véase Figura 6.12). Tradicionalmente, se ha abordado con algoritmos clásicos como máquinas de vectores de soporte o k -nearest neighbours [Kan15].

La irrupción del Aprendizaje Profundo, especialmente con redes neuronales convolucionales, supuso un gran avance [Oua17]. AlexNet, propuesta por A. Krizhevsky en [KSH17], fue pionera en esta revolución, logrando una mejora sustancial en el desafío ImageNet 2012, reduciendo el error del 26.2 % al 15.3 %.

En los años siguientes, surgieron modelos más sofisticados. VGG16, de Simonyan y Zisserman [SZ14], incrementó la profundidad de la red a 16 capas convolucionales con filtros de 3×3 , mejorando la precisión hasta un 7.3 % de error.

En paralelo, M. Lin et al. [LCY13] introdujeron el módulo *inception*, que permitió combinar múltiples convoluciones en paralelo. Esta idea fue aplicada por C. Szegedy et al. en GoogLeNet o Inception V1 [SLJ⁺14], optimizando el uso de parámetros y mejorando la eficiencia de la red.



Figura 6.12.: Resultado de aplicar clasificación de imágenes en una fotografía que contiene dos categorías: Caballo y Persona. Imagen extraída de [Sor18].

6.3.2. Detección de objetos

Esta tarea busca localizar objetos de distintas clases en una imagen y clasificarlos correctamente (véase Figura 6.13). Para ello, se utilizan cajas delimitadoras (*bounding boxes*) que encierran cada objeto.

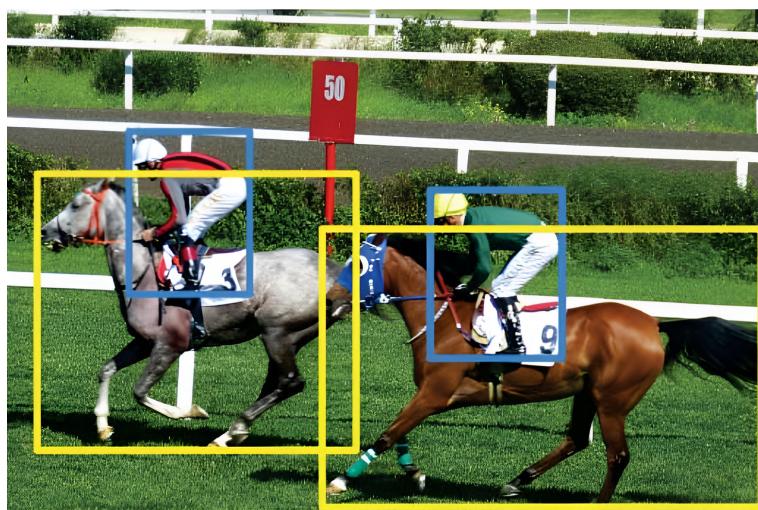


Figura 6.13.: Resultado de aplicar detección de objetos en una fotografía que contiene dos categorías: Caballo y Persona. Imagen extraída de [Sor18].

Las redes convolucionales han demostrado gran eficacia en esta tarea [GDDM13]. Sin embargo, su aplicación directa presenta retos: el número de objetos es variable y estos aparecen en distintas ubicaciones y tamaños, lo que hace inviable evaluar todas las regiones posibles

6. Fundamentos Teóricos

[Roh18].

Existen dos enfoques principales: métodos de una etapa y de dos etapas. Los de una etapa (YOLO [RDGF15], SSD [LAE⁺15], etc.) detectan y clasifican objetos en una única pasada, siendo rápidos pero menos precisos con objetos pequeños o tareas más complejas [Jor18a]. Los de dos etapas (R-CNN [GDDM13], Fast R-CNN [Gir15], Faster R-CNN [RHGS15]) primero generan regiones de interés y luego las clasifican, logrando mayor precisión.

R-CNN introdujo el uso de muestreo selectivo para generar 2000 regiones por imagen, clasificadas con SVM sobre vectores de características extraídos por una red convolucional. Aunque preciso, era lento tanto en entrenamiento como en inferencia [ZZXW18].

Fast R-CNN optimizó este proceso extrayendo primero el mapa de características completo y generando regiones sobre él, lo que redujo drásticamente el tiempo de inferencia [Gir15]. Finalmente, Faster R-CNN sustituyó el muestreo selectivo por una red de propuestas de regiones (RPN), integrando todo el proceso en un solo modelo entrenable de forma conjunta [RHGS15].

Hoy en día, los métodos de dos etapas siguen destacando por su alta precisión, siendo ampliamente utilizados en tareas de detección exigentes [Pin17].

6.3.3. Segmentación semántica

La segmentación semántica asigna una clase a cada píxel de la imagen, sin distinguir entre instancias (véase Figura 6.14).



Figura 6.14.: Resultado de aplicar segmentación semántica en una fotografía que contiene dos categorías: Caballo y Persona. Imagen extraída de [Sor18].

Inicialmente, esta tarea se abordó mediante clasificación por parches, pero el avance llegó con las redes completamente convolucionales (FCN) propuestas por J. Long et al. [LSD14],

que permitieron segmentar imágenes de tamaño arbitrario siguiendo una estructura de codificador-decodificador. Para recuperar la resolución perdida durante la reducción de escala, se introdujeron conexiones de salto entre capas [Jor18b].

Posteriormente, O. Ronneberger et al. desarrollaron U-Net [RFB15], una variante simétrica de las FCN optimizada para segmentación médica, donde las conexiones de salto realizan concatenaciones en lugar de sumas, mejorando la precisión espacial.

La mayoría de los modelos actuales siguen esta estructura, diferenciándose principalmente en los bloques empleados. M. Drozdzal et al. [DVC⁺16] incorporaron bloques residuales para acelerar la convergencia, mientras que S. Jegou et al. [JDV⁺16] propusieron bloques densos para un mejor aprovechamiento de las características de bajo nivel.

6.3.4. Segmentación de instancias

La segmentación de instancias combina las dos tareas anteriores de Visión por Computador: la detección de objetos y la segmentación semántica. Su objetivo es asignar a cada píxel una clase, al tiempo que diferencia entre múltiples instancias de una misma clase (véase Figura 6.15).



Figura 6.15.: Resultado de aplicar segmentación de instancias en una fotografía que contiene dos categorías: Caballo y Persona. Imagen extraída de [Sor18].

Entre las primeras propuestas, destaca el trabajo de Hariharan et al. [HAGM14b], quienes extendieron R-CNN para generar una máscara por instancia, utilizando propuestas de regiones y redes neuronales para su clasificación. Posteriormente, introdujeron el concepto de *hipercolumnas* [HAGM14a], mejorando la precisión espacial al combinar activaciones de múltiples capas.

Otros enfoques como DeepMask [PCD15] e InstanceFCN [DHL⁺16] buscaron reducir el número de propuestas y mejorar la eficiencia computacional. Li et al. [LQD⁺16] dieron un

6. Fundamentos Teóricos

paso más con un modelo totalmente convolucional y de extremo a extremo, que combinaba segmentación y detección en un solo marco.

También se propusieron estrategias alternativas basadas en segmentación semántica seguida de separación de instancias, como InstanceCut [KLA⁺16], técnicas basadas en la transformación de cuencas [BU16] o redes de agrupamiento secuencial [LJFU17]. Aunque innovadoras, estos enfoques presentan limitaciones ante occlusiones, objetos pequeños o estructuras no conectadas.

El mayor avance llegó con Mask R-CNN [HGDG17], una extensión de Faster R-CNN que incorpora una rama adicional para predecir máscaras por instancia además de las cajas delimitadoras y las clases. Esta arquitectura alcanzó el estado del arte en el conjunto COCO [LMB⁺14] y será explicada con mayor detalle más adelante.

6.4. 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.

6.4.1. 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.

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

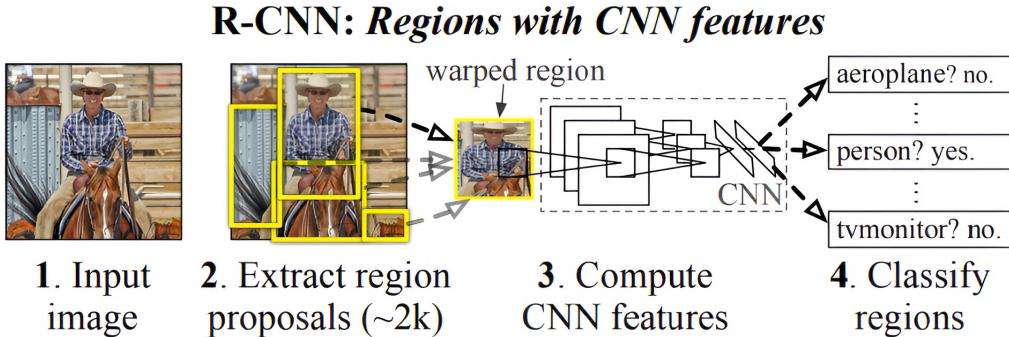


Figura 6.16.: Arquitectura R-CNN. Imagen extraída de [GDDM13].

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.

6.4.2. 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 Layer* 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 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.

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.

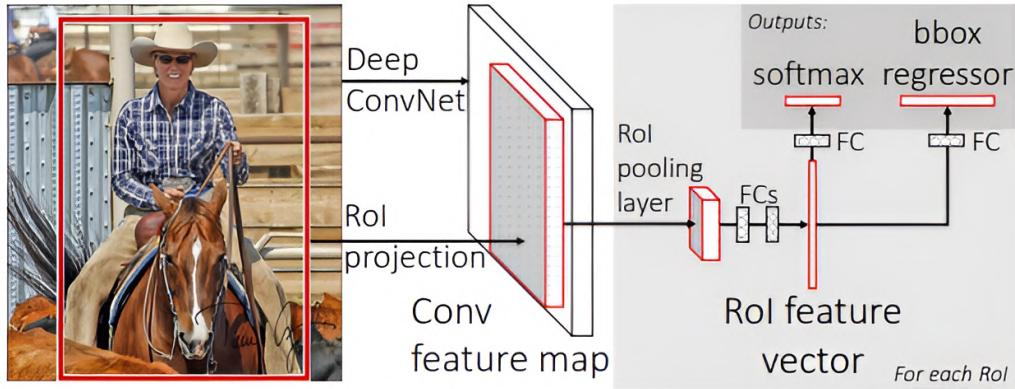


Figura 6.17.: Arquitectura Fast R-CNN. Imagen extraída de [Gir15].

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.

6.4.3. 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.

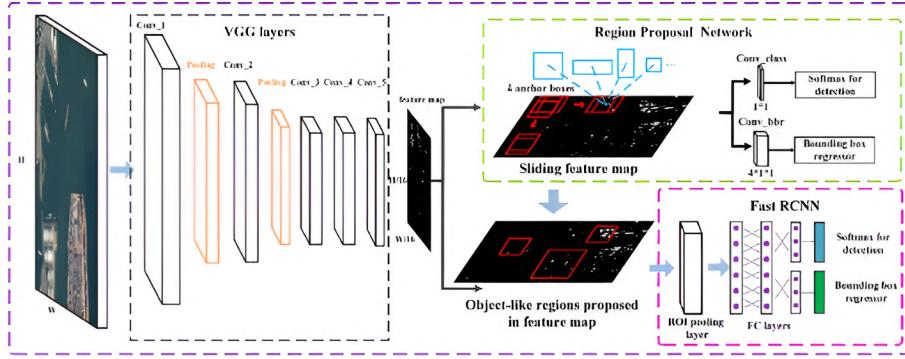


Figura 6.18.: Arquitectura Faster R-CNN. Imagen extraída de [DSZ⁺18].

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.
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.

6.5. Inteligencia Artificial Explicable

La XAI es una disciplina emergente dentro de la IA 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, han ampliado sus aplicaciones a ámbitos críticos 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.

6.5.1. Conceptos fundamentales

La motivación detrás de la XAI radica en la necesidad de transparencia en los sistemas de IA, 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

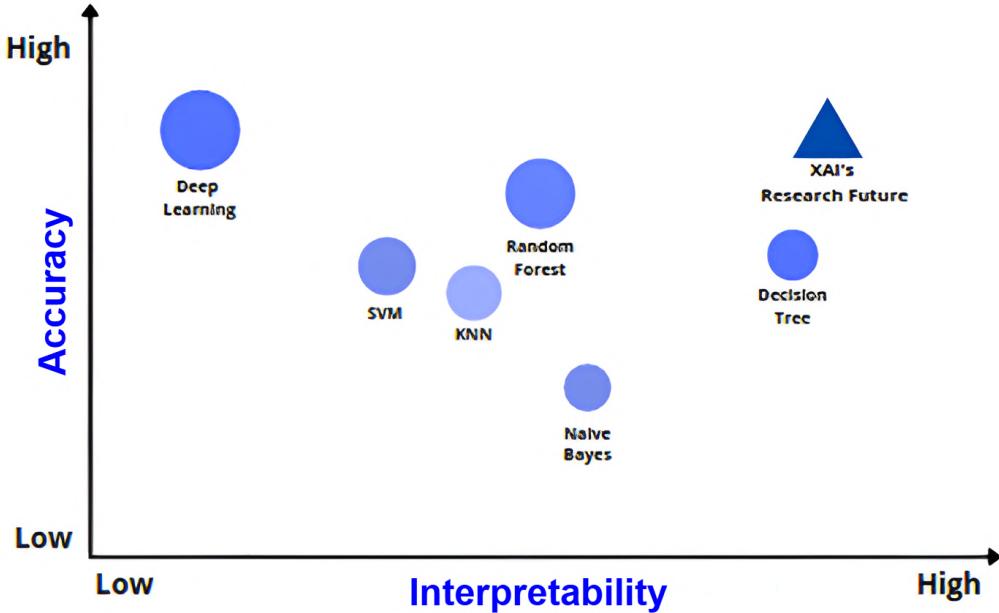


Figura 6.19.: Exactitud frente a interpretabilidad para distintos algoritmos de Aprendizaje Automático. Imagen extraída de [ASJ⁺21].

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 [DVK17a].

6.5.2. 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, según la literatura [AB18, ADRS⁺19], en función de dos criterios fundamentales: el momento en que se aplica la interpretación y el alcance de la misma. A continuación, se describen ambos criterios con mayor detalle:

- **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 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.

6.5.3. Propiedades deseables

Para que un método de explicabilidad sea efectivo, debe cumplir con ciertas propiedades:

- **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.

Estas propiedades han sido ampliamente discutidas en la literatura sobre explicabilidad. En particular, la fidelidad y la estabilidad han sido formuladas en trabajos como [MSM18],

6. Fundamentos Teóricos

mientras que la comprensibilidad y la justicia se fundamentan en los enfoques de interpretabilidad orientados al usuario y a la equidad, como se plantea en [Fre14, DVK17b, MRW19].

En este trabajo, la evaluación de estas propiedades se aborda de la siguiente manera: la fidelidad y la estabilidad se cuantifican mediante las métricas Deletion, Insertion, Stability e IoU_{Grad-CAM}, presentadas en la Subsección 10.3.3. La comprensibilidad se examina a través de la interpretación visual de los mapas Grad-CAM generados, prestando atención a la claridad y localización de las regiones activadas. Por último, la justicia se analiza mediante la comparación de las métricas explicativas entre diferentes categorías dentales, con el objetivo de identificar posibles disparidades o sesgos en las explicaciones proporcionadas por el modelo.

6.5.4. 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 parte de un modelo ya entrenado y permite generar explicaciones a posteriori sobre su comportamiento, mediante distintos métodos que pueden centrarse en atribuir relevancia a las características de entrada, contrastar decisiones con ejemplos alternativos, o sintetizar prototipos representativos.

A continuación, estructuraremos las principales técnicas *post-hoc* en redes neuronales convolucionales en tres grandes grupos: métodos basados en eliminación, en gradientes y en propagación. Esta organización se inspira en el manifiesto propuesto por Longo et al. [LBC⁺24] sobre los desafíos abiertos en XAI.

6.5.4.1. 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, consideraremos 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 [LL17].

6.5.4.2. 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 retropropagación (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 Prominencia (*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 para identificar qué segmentos de una imagen son cruciales para una clasificación específica [SVZ13].
- **Retropropagación Guiada (*Guided Backpropagation*):** Consiste en una variante del método de retropropagación, 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 prominencia 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].

6.5.4.3. 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

6. Fundamentos Teóricos

de algoritmos clásicos, como la retropropagación, 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

Este capítulo ofrece una revisión detallada de la literatura científica más reciente en dos ámbitos clave: las arquitecturas avanzadas de redes neuronales para segmentación de instancias, y los enfoques actuales de XAI aplicados para la arquitectura Mask R-CNN. Para ello, se ha llevado a cabo una búsqueda sistemática en la base de datos Scopus, con el objetivo de identificar las contribuciones más relevantes que fundamentan y contextualizan el presente trabajo.

7.1. Segmentación de imágenes dentales

La segmentación automática de estructuras dentales en radiografías panorámicas representa un área de creciente interés en el ámbito de la Visión por Computador aplicada a la odontología. Para explorar el estado actual de la investigación en este campo, se realizó una búsqueda bibliográfica en la base de datos Scopus.

La primera búsqueda, empleando la consulta:

```
TITLE-ABS-KEY ( ( "dental" AND "segmentation" ) ),
```

arrojó un total de 1992 resultados, lo que refleja el amplio volumen de publicaciones relacionadas con la segmentación dental en general (véase [Figura 7.1](#)). Este crecimiento sostenido se hace especialmente evidente a partir del año 2015, coincidiendo con el auge de las técnicas basadas en aprendizaje profundo.

Para acotar el análisis a investigaciones que utilizan la arquitectura Mask R-CNN, se refinó la búsqueda con los términos:

```
TITLE-ABS-KEY ( ( "dental" AND "segmentation" ) AND  
                  ( "mask" AND "r-cnn" ) ),
```

obteniéndose 37 publicaciones (véase [Figura 7.2](#)). Esta cifra revela que el uso específico de Mask R-CNN en el ámbito dental constituye aún una línea emergente de investigación, con un número creciente de contribuciones en los últimos años.

En el ámbito del procesamiento de imágenes médicas, la segmentación automática de estructuras dentales en ortopantomografías ha emergido como una línea de investigación de especial relevancia. Estas imágenes panorámicas, ampliamente utilizadas en odontología, permiten una visualización integral del arco dentario, pero presentan notables desafíos técnicos derivados del solapamiento de estructuras, la baja calidad del contraste y la presencia frecuente de artefactos. Ante estas dificultades, las soluciones basadas en redes neuronales profundas han ganado protagonismo por su capacidad para automatizar y mejorar la precisión en tareas como la segmentación y numeración dental.

7. Estado del Arte

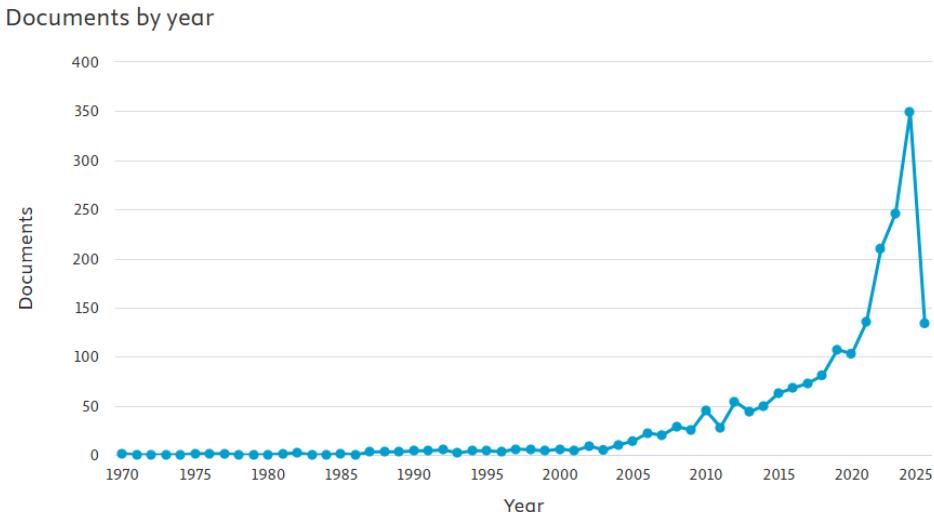


Figura 7.1.: Número de publicaciones por año obtenidas en Scopus con la búsqueda sobre segmentación dental, realizada el 25 de abril de 2025. Se identificaron un total de 1992 documentos, lo que evidencia un crecimiento sostenido en el interés por la segmentación en imágenes dentales, especialmente a partir del año 2015. Imagen extraída de [Sco24].

Entre las arquitecturas más destacadas para abordar esta tarea se encuentra Mask R-CNN, un modelo ampliamente utilizado en segmentación de instancias por su capacidad para detectar, clasificar y segmentar objetos individuales en una única red. A continuación, se presentan algunos de los trabajos más relevantes identificados en la búsqueda bibliográfica específica sobre segmentación dental con Mask R-CNN, los cuales guardan estrecha relación con el enfoque y objetivos del presente trabajo.

Los primeros avances significativos en este campo llegaron en 2020, cuando Lee et al. [LHK⁺20] desarrollaron un enfoque completamente automatizado para la segmentación dental en imágenes panorámicas, empleando una red Mask R-CNN entrenada sobre 846 imágenes anotadas manualmente por un radiólogo oral. Mediante técnicas de aumento de datos, ampliaron el conjunto de entrenamiento a 1024 imágenes, lo que contribuyó a mejorar la robustez del modelo y evitar el sobreajuste. El sistema fue validado sobre 20 imágenes adicionales, obteniendo un F1-score de 0.875 y un IoU medio de 0.877. La evaluación visual confirmó una alta concordancia con las segmentaciones manuales, demostrando que Mask R-CNN podía adaptarse con éxito al entorno clínico dental, incluso ante las dificultades propias de las radiografías panorámicas.

En esa misma línea, Silva et al. [SPOP20] propusieron ese mismo año un análisis comparativo entre varias arquitecturas de segmentación por instancias, incluyendo Mask R-CNN, PANet, Hybrid Task Cascade (HTC) y ResNeSt, sobre una versión ampliada del conjunto de datos UFBA-UESC Dental Images. A diferencia de trabajos anteriores que abordaban la segmentación y numeración por separado, su propuesta contempló ambas tareas de forma unificada. Aunque todas las redes evaluadas ofrecieron buenos resultados, PANet se destacó con un mAP del 71.3 %, consolidando así el valor de las arquitecturas profundas en tareas

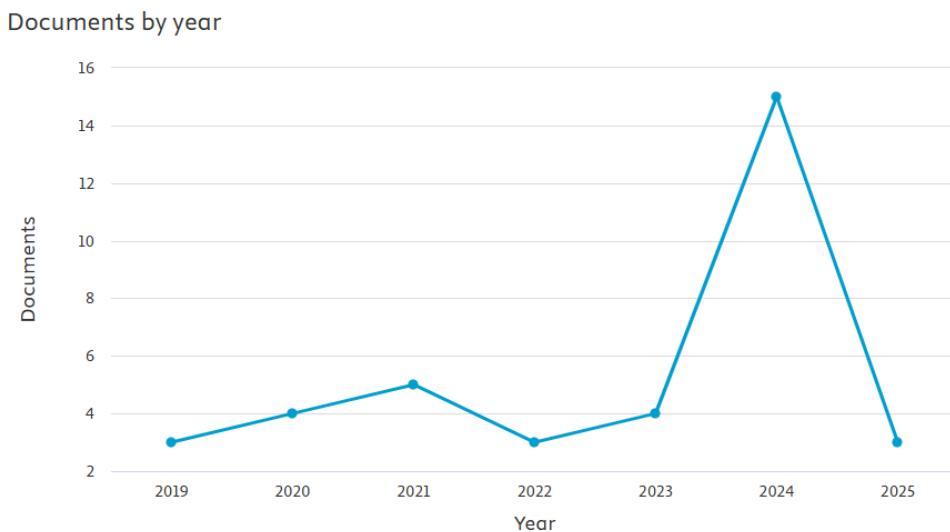


Figura 7.2.: Número de publicaciones por año obtenidas en Scopus con la búsqueda sobre segmentación dental con Mask R-CNN, realizada el 25 de abril de 2025. El número total de documentos encontrados fue de 37, lo que indica que la aplicación específica de Mask R-CNN a imágenes dentales representa una línea de investigación más reciente y aún en desarrollo. Imagen extraída de [Sco24].

complejas como la segmentación dental multiclase.

A medida que esta línea de investigación maduraba, comenzaron a surgir estudios centrados en conjuntos de datos más grandes y detallados. Rubiu et al. [RBC⁺23], en 2023, desarrollaron un modelo Mask R-CNN utilizando el Tufts Dental Database, compuesto por 1000 radiografías panorámicas etiquetadas con 52 clases de dientes deciduos y permanentes. Tras un entrenamiento de 300 épocas con aumento de datos, el modelo fue evaluado en un conjunto de prueba de 50 imágenes, alcanzando una precisión del 98.4 % y un Dice Similarity Coefficient medio de 0.87. Estos resultados, especialmente sólidos en incisivos mandibulares y molares, subrayaron la capacidad del modelo para identificar y segmentar dientes individuales con gran precisión, incluso en casos de dentición mixta. Además, los autores destacaron el potencial clínico de esta herramienta para asistir a profesionales menos experimentados, reduciendo la variabilidad interobservador y acelerando los flujos de trabajo.

En paralelo a estos estudios basados en conjuntos de datos públicos, otros autores decidieron construir sus propias bases de datos ante la escasez de conjuntos accesibles y bien anotados. Brahmi e Imen Jdey [BJ23] desarrollaron en 2023 una arquitectura Mask R-CNN aplicada sobre un nuevo conjunto de 107 imágenes obtenidas en clínicas odontológicas. Esta contribución no solo proporcionó un nuevo conjunto de datos real y anotado, sino que permitió validar el rendimiento del modelo en un entorno clínico real. El sistema alcanzó un mAP del 90 %, una precisión del 96 % y un F1-score del 63 %, evidenciando una alta capacidad de localización, aunque con margen de mejora en la recuperación global de instancias. Aun con estas limitaciones, el estudio refuerza el papel de Mask R-CNN como una herramienta fiable para el análisis automatizado de imágenes dentales y para el soporte al diagnóstico asistido por IA.

7. Estado del Arte

En conjunto, estos trabajos delinean una evolución clara en el uso de Mask R-CNN para la segmentación dental: desde enfoques pioneros sobre conjuntos de datos limitados, pasando por modelos multiclas con altos niveles de precisión, hasta propuestas validadas en entornos clínicos reales. Esta trayectoria sugiere que las redes de segmentación por instancias no solo son viables en el dominio odontológico, sino que están cada vez más cerca de integrarse en la práctica clínica diaria como asistentes inteligentes para tareas de planificación, diagnóstico y documentación.

Para completar esta revisión, la [Tabla 7.1](#) recoge un resumen comparativo de los trabajos analizados que utilizan Mask R-CNN para la segmentación dental en radiografías panorámicas. Se presentan los autores, el título, el año de publicación, el número de citas (según Scopus) y una breve descripción de las principales contribuciones y métricas de rendimiento más relevantes reportadas por cada estudio.

7.2. Técnicas de explicabilidad aplicadas a Mask R-CNN

A medida que las arquitecturas de segmentación profunda como Mask R-CNN ganan protagonismo en aplicaciones críticas, aumenta también la necesidad de comprender mejor su funcionamiento interno. En este contexto, las técnicas de explicabilidad e interpretabilidad buscan proporcionar mecanismos que permitan visualizar o justificar las decisiones de modelos complejos, aumentando la transparencia y la confianza en su uso.

Para explorar el estado actual de la investigación en este ámbito, se realizó de nuevo una búsqueda bibliográfica en la base de datos Scopus. La primera consulta general:

```
TITLE-ABS-KEY ( ( ("mask" AND "r-cnn") AND  
 ( ("explainable" AND "ai") OR "interpretability"  
 OR "explainability" OR "xai" ) ) ),
```

centrada en trabajos que aplican técnicas de explicabilidad a Mask R-CNN, arrojó un total de 35 resultados (véase [Figura 7.3](#)), lo que evidencia un creciente interés en esta intersección entre modelos de segmentación y XAI.

Al refinar la búsqueda con el términos específico Grad-CAM:

```
TITLE-ABS-KEY ( ( ("mask" AND "r-cnn") AND ( "grad-cam" OR "gradcam" ),
```

se identificaron 11 publicaciones que aplican directamente esta técnica de visualización a Mask R-CNN (véase [Figura 7.3](#)). Esta cifra más reducida revela que la integración de Grad-CAM en entornos de segmentación por instancias aún constituye una línea emergente, con mucho margen para la innovación.

Entre los estudios más relevantes identificados se encuentran tres contribuciones del grupo de Xavier Alphonse Inbaraj y colaboradores, quienes han propuesto diversas combinaciones entre Grad-CAM y Mask R-CNN con el objetivo de mejorar la localización, interpretación y

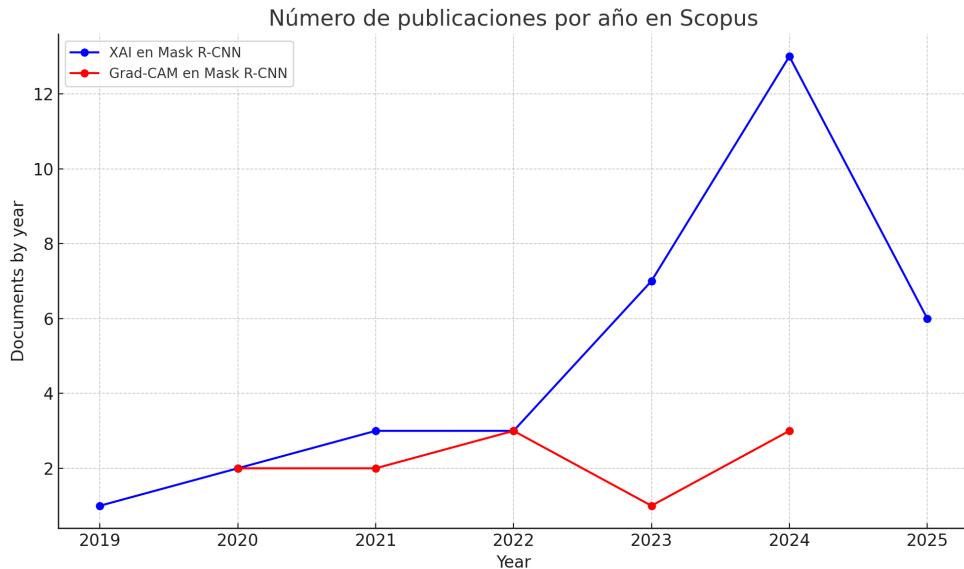


Figura 7.3.: Número de publicaciones por año obtenidas en Scopus con la búsqueda sobre técnicas de explicabilidad aplicadas a Mask R-CNN (línea azul) y sobre la aplicación específica de Grad-CAM a Mask R-CNN (línea roja), realizada el 4 de mayo de 2025. Se identificaron un total de 35 documentos en la primera búsqueda y 11 en la segunda. Estos resultados evidencian un creciente interés por dotar de explicabilidad a arquitecturas como Mask R-CNN, especialmente en los últimos años, y sitúan a Grad-CAM como una línea de investigación aún en fase inicial. Imagen extraída de [Sco24].

rendimiento de los modelos.

En primer lugar, Inbaraj et al. [XJ21] propusieron la arquitectura Mask-GradCAM, que integra Grad-CAM++ con Mask R-CNN con el objetivo de mejorar la visualización e interpretabilidad de las predicciones en tareas de detección y segmentación de instancias. El enfoque se basa en generalizar la técnica Grad-CAM para producir mapas de activación de alta resolución, generando una representación visual más precisa de las regiones relevantes para el modelo sin necesidad de modificar su arquitectura base. El sistema se apoya en una red *backbone* ResNet-101 combinada con *Feature Pyramid Networks* (FPN), y emplea una estrategia de fusión de mapas de activación mediante multiplicación píxel a píxel entre la imagen y el heatmap generado. Como componente innovador, introducen una cadena de procesos denominada *mask chain cascading*, compuesta por cuatro etapas: generación de anclas, afinamiento de características, predicción de máscaras y categorización por instancia, cada una optimizada mediante funciones matemáticas diferenciables que permiten retropropagación. En la evaluación experimental, el modelo fue probado sobre el conjunto de datos COCO, mostrando mejoras en precisión visual, capacidad de localización y generación de mapas de atención respecto a Grad-CAM y Grad-CAM++. En particular, Mask-GradCAM logró una precisión de entrenamiento del 93 % y de test del 94.5 %, superando en más del 13 % los resultados obtenidos por técnicas de base como Grad-CAM, lo que evidencia su eficacia en la explicación de decisiones en arquitecturas de segmentación complejas.

7. Estado del Arte

En una segunda contribución, Inbaraj et al. [XVM⁺₂₁] desarrollaron la arquitectura GC-MRCNN (*Grad-CAM++ with Mask Regional Convolutional Neural Network*), cuyo objetivo es mejorar la identificación y localización de objetos, especialmente en entornos con occlusiones, ruido o variaciones de escala. Esta propuesta integra la capacidad de explicación visual de Grad-CAM++ con la potencia de detección y segmentación de Mask R-CNN, empleando una red ResNet-101 como *backbone* y el conjunto de datos COCO para su evaluación. A nivel metodológico, el sistema genera mapas de activación de alta resolución mediante Grad-CAM++ y refuerza la localización mediante Mask R-CNN, aprovechando el alineamiento preciso de regiones de interés y la producción de máscaras a nivel de píxel. La arquitectura permite mantener la estructura original del modelo, evitando modificaciones complejas, y emplea estrategias como el aprendizaje escalonado y la poda de cajas redundantes para optimizar el rendimiento. Los resultados experimentales mostraron una precisión del 82 % y mejoras significativas frente a otros métodos, con un aumento del 17.02 % en la tasa de detección respecto a Grad-CAM y valores de AP superiores (AP: 49.57, AP@50: 71.88, AP@75: 65.25), evidenciando la eficacia del enfoque para explicar y mejorar las predicciones del modelo. Estas mejoras se atribuyen a la combinación efectiva entre mapas de atención y segmentación por instancias, lo que permite localizar objetos incluso en situaciones con múltiples elementos de la misma clase o presencia de clases solapadas.

En su tercer trabajo, Inbaraj et al. [XVM⁺₂₂] propusieron el modelo GradCAM-MLRCNN, que combina la técnica de visualización Grad-CAM++ con la arquitectura Mask R-CNN y algoritmos de aprendizaje automático tradicionales para mejorar la detección de objetos. Esta propuesta integra mapas de activación generados por Grad-CAM++ para localizar regiones relevantes y luego utiliza Mask R-CNN para detectar objetos en dichas regiones, complementando la clasificación con modelos como regresión logística, árboles de decisión o vecinos más cercanos. A nivel metodológico, las imágenes se procesan para generar mapas de calor que guían la detección de objetos, los cuales son posteriormente clasificados mediante algoritmos supervisados. El sistema se evaluó utilizando el conjunto de datos MS COCO y varios extractores de características preentrenados (VGG16, VGG19, ResNet101 y ResNet152), destacando la combinación de ResNet152 con regresión logística, que alcanzó un accuracy del 98.4 %, un recall del 99.6 % y una precision del 97.3 %. Además, los autores validaron estadísticamente su enfoque mediante una prueba chi-cuadrado, mostrando una relación significativa entre las predicciones correctas y erróneas. Este trabajo evidencia el potencial de combinar mapas de activación con técnicas clásicas de clasificación para mejorar tanto la explicabilidad como el rendimiento en tareas de segmentación por instancias.

En conjunto, estas tres contribuciones representan una evolución metodológica significativa en la aplicación de técnicas de explicabilidad a Mask R-CNN. Desde la mejora visual de mapas de atención sin alterar la arquitectura base (Mask-GradCAM), pasando por el refuerzo de la localización en escenarios complejos mediante estrategias de optimización (GC-MRCNN), hasta la integración de algoritmos de aprendizaje automático tradicionales para complementar la segmentación con una clasificación más robusta (GradCAM-MLRCNN), los trabajos del grupo de Inbaraj muestran distintas vías para dotar de mayor transparencia y rendimiento a modelos de segmentación por instancias.

La [Tabla 7.1](#) recoge un resumen comparativo de los estudios analizados en este capítulo, detallando autores, título, año, número de citas (según Scopus) y principales contribuciones.

7.2. Técnicas de explicabilidad aplicadas a Mask R-CNN

Autores	Título	Año	Citas	Principales contribuciones
Lee et al. [LHK ⁺ 20]	<i>Application of a fully deep convolutional neural network to the automation of tooth segmentation on panoramic radiographs</i>	2020	138	Aplicación de Mask R-CNN con <i>fine-tuning</i> sobre un conjunto de datos anotado manualmente. Evaluación visual y cuantitativa con alta concordancia respecto al <i>ground truth</i> .
Silva et al. [SPOP20]	<i>A study on tooth segmentation and numbering using end-to-end deep neural networks</i>	2020	63	Benchmark exhaustivo de redes profundas para segmentación, detección y numeración dental en radiografías panorámicas. Evaluación sobre UFBA-UESC Deep con cuatro arquitecturas.
Inbaraj et al. [XJ21]	<i>Mask-GradCAM: Object Identification and Localization of Visual Presentation for Deep Convolutional Network</i>	2021	4	Generalización de Grad-CAM++ para obtener mapas de atención de alta resolución sin modificar la arquitectura. Introducción del proceso <i>mask chain cascading</i> para mejorar la segmentación por etapas diferenciables.
Inbaraj et al. [XVM ⁺ 21]	<i>Object Identification and Localization Using Grad-CAM++ with Mask Regional Convolution Neural Network</i>	2021	19	Refuerzo de la localización y detección en entornos con occlusiones y ruido. Integración directa de mapas Grad-CAM++ y Mask R-CNN con técnicas de poda y aprendizaje escalonado.
Inbaraj et al. [XVM ⁺ 22]	<i>Object Detection via Gradient-Based Mask R-CNN Using Machine Learning Algorithms</i>	2022	12	Combinación de Grad-CAM++ y Mask R-CNN con clasificadores clásicos (regresión logística, árboles, KNN). Validación estadística y mejora de precisión mediante fusión de predicciones.
Rubiu et al. [RBC ⁺ 23]	<i>Teeth segmentation in panoramic dental X-ray using Mask-RCNN</i>	2023	19	Segmentación automática de dientes permanentes y deciduos en radiografías panorámicas mediante Mask R-CNN entrenada con el conjunto de datos Tufts.
Brahmi y Jdey [BJ23]	<i>Automatic tooth instance segmentation and identification from panoramic X-Ray images using deep CNN</i>	2024	8	Propuesta de un sistema automático de segmentación e identificación dental con Mask R-CNN entrenado en un conjunto de datos propio.

Tabla 7.1.: Resumen de estudios relevantes que utilizan Mask R-CNN para segmentación dental, incluyendo investigaciones que integran técnicas de explicabilidad como Grad-CAM. Se muestran los autores, título, año, número de citas (según Scopus) y principales contribuciones.

8. Métodos Empleados

En este capítulo se detallarán los métodos empleados 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 (véase Figura 8.1). La arquitectura de Mask R-CNN se muestra en la Figura 8.2 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).
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 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.

8. Métodos Empleados

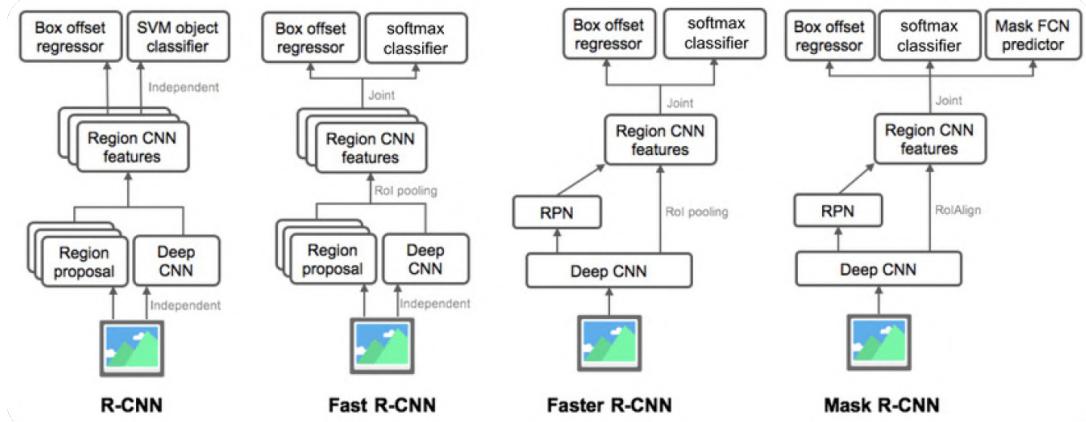


Figura 8.1.: Evolución de la familia R-CNN: de R-CNN a Mask R-CNN. Cada versión introduce mejoras en velocidad, integración y capacidades. En particular, Mask R-CNN añade dos componentes destacados respecto a Faster R-CNN: una nueva rama de salida que predice máscaras binarizadas por instancia (*Mask FCN predictor*) y la sustitución de *RoI Pooling* por *RoI Align*. Imagen extraída de [Vin24].

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

El aumento de la profundidad en las redes neuronales convolucionales ha permitido extraer características cada vez más complejas. Sin embargo, esto también conlleva problemas como la desaparición del gradiente o la degradación del rendimiento. Para abordar estos desafíos, He et al. propusieron la arquitectura ResNet [HZRS15], basada en la idea de aprendizaje residual.

La principal innovación de ResNet consiste en introducir conexiones de salto (skip connections) que permiten a la red aprender una función residual $F(x)$ respecto a la entrada x , de modo que la salida se expresa como $H(x) = F(x) + x$. Este diseño facilita la propagación del gradiente y permite entrenar redes mucho más profundas sin pérdida de rendimiento.

Cada bloque residual puede incluir una o varias capas convolucionales, manteniendo la conexión directa entre entrada y salida. Esta estrategia ha demostrado mejorar significativamente la convergencia y la precisión en tareas de visión por computador. Debido a su robustez y rendimiento, ResNet se utiliza habitualmente como backbone en arquitecturas avanzadas como Mask R-CNN.

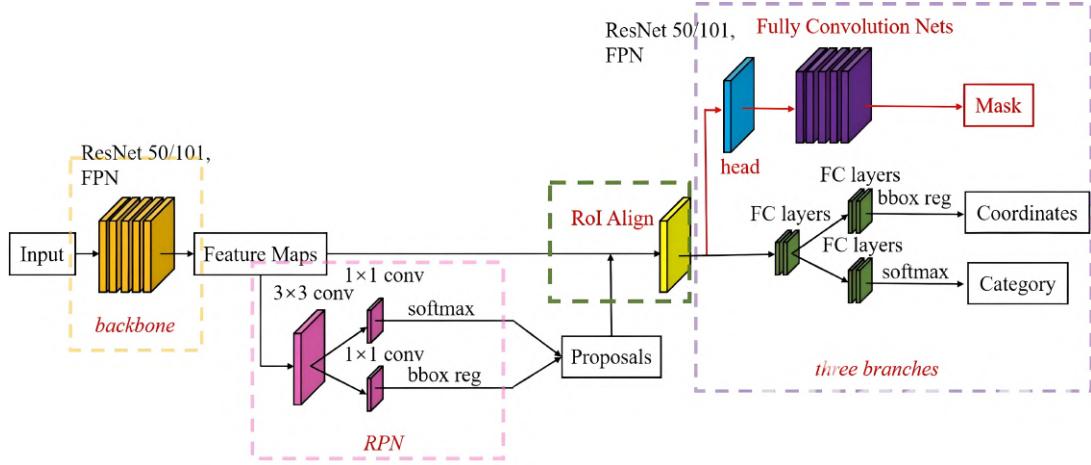


Figura 8.2.: Arquitectura de Mask R-CNN, una extensión de Faster R-CNN que incorpora una rama adicional para la predicción de máscaras. El flujo comienza con la imagen de entrada, que es procesada por una red convolucional profunda (ResNet-50/101 con FPN, en amarillo) para obtener mapas de características multiescalares (*backbone*). A continuación, la Red de Propuestas de Regiones (RPN, en rosa) genera posibles regiones de interés (RoIs) mediante convoluciones y clasificadores softmax y regresores de cajas delimitadoras. Las RoIs generadas se alinean espacialmente con la operación *RoI Align* (en verde), que elimina errores de cuantización presentes en métodos anteriores. Cada RoI alineada se procesa a través de tres ramas principales (en morado): (1) una para clasificación (softmax), (2) otra para regresión de coordenadas de cajas (*bbox reg*) y (3) una rama adicional convolucional para predecir máscaras de segmentación por píxel. Imagen extraída de [SKY23].

8.1.1.2. Red piramidal de características

Como se muestra en la Figura 8.3, 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 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

8. Métodos Empleados

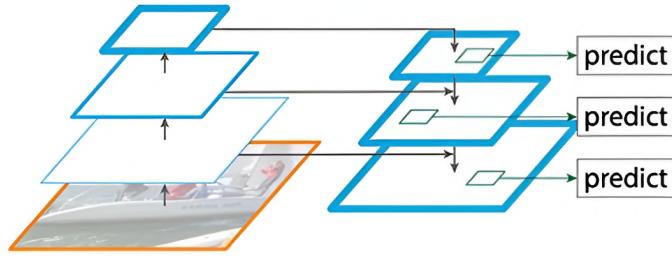


Figura 8.3.: El *backbone* que utiliza la estructura FPN está formado por rutas ascendentes y descendentes conectadas mediante conexiones de salto. Imagen extraída de [LDG⁺16].

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 8.4, $(i + u, j + v)$ 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.

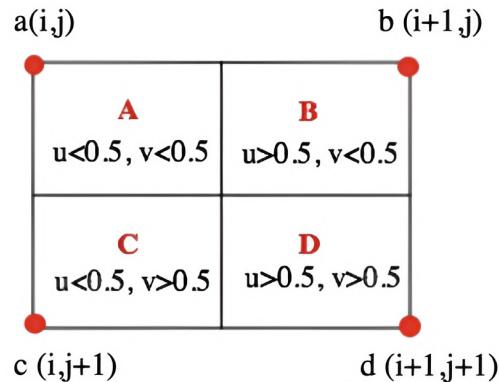


Figura 8.4.: Ejemplo de interpolación por vecino más cercano.

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, proporcionando distintas resoluciones. Antes de comenzar a trabajar la Red de Propuestas de Regiones, la cual calculará las regiones de interés, 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 8.5).

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. Métodos Empleados

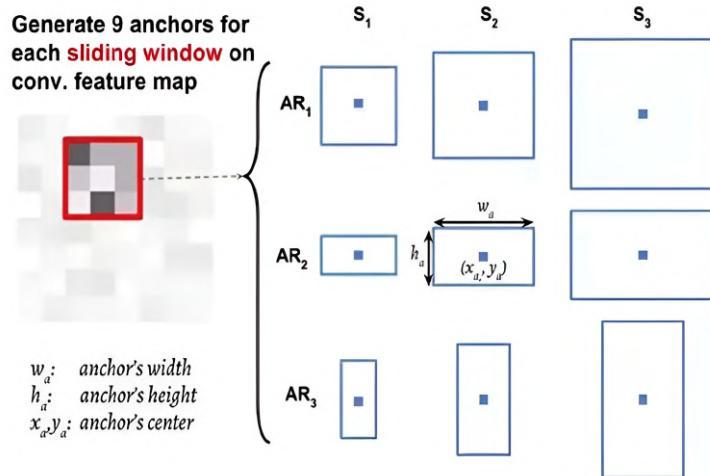


Figura 8.5.: Las distintas *anchor boxes* generadas tienen diferentes escalas y relaciones de aspecto. Imagen extraída de [Bup18].

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. 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*. Segundo, 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 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 8.6](#) muestra la diferencia entre ambas capas.

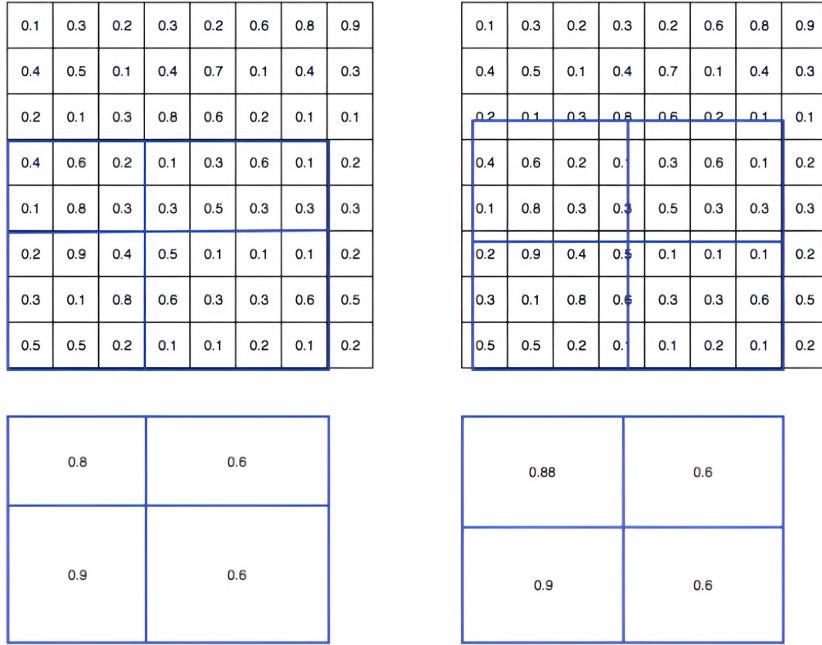


Figura 8.6.: Comparación entre *RoI Pooling* (izquierda) y *RoI Align* (derecha). Imagen extraída de [[Hui18a](#)].

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 a la RPN, con dos diferencias principales. En primer lugar, no 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 8.7](#) se observa un ejemplo de resultados obtenidos por Mask R-CNN para el conjunto de datos COCO [[LMB⁺14](#)].

8. Métodos Empleados



Figura 8.7.: 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].

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:

- 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 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:

$$\mathcal{L}_{\text{cls}} = -\frac{1}{N} \sum_{i=1}^N \log(p_i^c),$$

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* ($\mathcal{L}_{\text{bbox}}$):** 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:

$$\mathcal{L}_{\text{bbox}} = \sum_{i \in \{x,y,w,h\}} \text{smooth}_{L1}(\hat{t}_i - t_i),$$

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

8. Métodos Empleados

entrada que tienen una mayor influencia en la salida del modelo. A continuación, describimos detalladamente su funcionamiento.

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 8.8](#), se muestra un ejemplo ilustrativo del funcionamiento de Grad-CAM aplicado a problemas de clasificación. Esta técnica fue originalmente diseñada para redes convolucionales entrenadas con fines clasificatorios, y permite visualizar mediante mapas de calor las regiones de una imagen que más influyen en la predicción de una clase concreta. A través del análisis de los gradientes en relación con los mapas de características de capas convolucionales, Grad-CAM revela qué zonas activan con mayor intensidad la decisión final del modelo. Como se observa en la figura, la calidad y precisión de estas visualizaciones varía según la profundidad de la capa seleccionada, siendo las capas más profundas las que capturan conceptos semánticos más abstractos y, por tanto, más relevantes para la tarea de clasificación [SCD⁺17].

En la siguiente sección, se explora cómo esta técnica ha sido adaptada a tareas más complejas como la segmentación de instancias, integrándola con arquitecturas como Mask R-CNN.

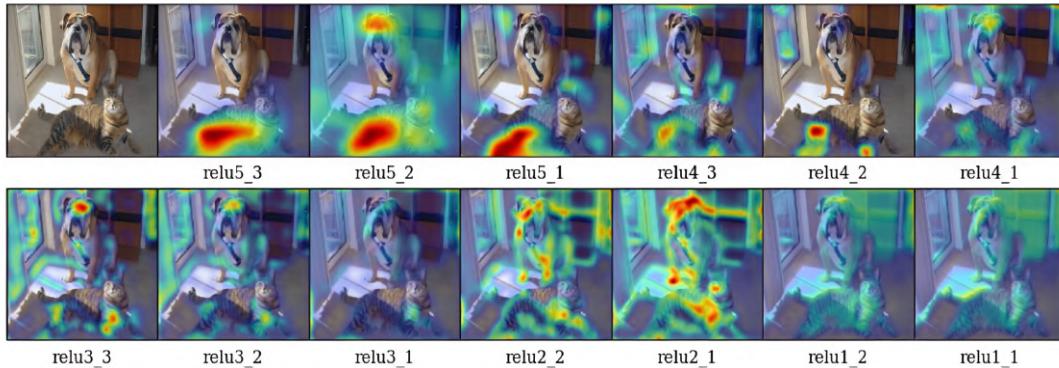


Figura 8.8.: Grad-CAM aplicado en diferentes capas convolucionales para la clase *tiger cat*.
 Esta figura examina cómo las localizaciones varían cualitativamente al aplicar Grad-CAM en distintos mapas de características de una red neuronal convolucional (VGG16 [SZ14]). Observamos que las visualizaciones 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. Métodos Empleados

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:

- 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.

8.3. Adaptación de Grad-CAM a Mask R-CNN

Como ya se ha comentado, la arquitectura original de Grad-CAM fue diseñada para tareas de clasificación en redes neuronales convolucionales estándar, donde se dispone de una única salida de clase por imagen. Sin embargo, en Mask R-CNN el flujo de datos es más complejo, ya que las predicciones se realizan de forma localizada sobre múltiples regiones de interés, cada una de las cuales puede estar asociada a una clase distinta. Por tanto, para aplicar Grad-CAM en este contexto es necesario realizar ciertas modificaciones que permitan obtener un mapa de atención asociado a una detección concreta. Diversos estudios han demostrado la viabilidad y utilidad de adaptar Grad-CAM a arquitecturas como Mask R-CNN, lo que permite visualizar no solo las regiones que influyen en la predicción de clase, sino también aquellas que intervienen en la generación de las máscaras segmentadas. Una revisión detallada del estado del arte sobre estas adaptaciones se ha presentado ya en la [Sección 7.2](#).

Aunque las máscaras generadas por modelos como Mask R-CNN ya representan explícitamente la segmentación de instancias, la aplicación de técnicas como Grad-CAM aporta una capa adicional de interpretabilidad que va más allá de la salida supervisada del modelo. En particular, Inbaraj et al. [[XJ21](#), [XVM⁺21](#), [XVM⁺22](#)], destacan que la combinación de Grad-CAM con Mask R-CNN permite generar mapas de activación que reflejan las regiones internas de la red que han influido más en la predicción, no simplemente la máscara final de salida. Esta visualización resulta útil para analizar si el modelo ha tomado su decisión a partir de regiones coherentes con la semántica del objeto, o si, por el contrario, ha basado su predicción en artefactos o zonas irrelevantes. Como señala Inbaraj et al., esta técnica permite validar que la red está mirando donde debería. Esto es especialmente relevante en presencia de múltiples objetos de la misma clase o condiciones complejas como ocultación o ruido [[XVM⁺21](#)]. Por tanto, aunque pueda haber coincidencias visuales entre la máscara segmentada y el mapa de Grad-CAM, ambas representan aspectos distintos del comportamiento del modelo: la primera, su salida supervisada; la segunda, su proceso interno de decisión.

8.3. Adaptación de Grad-CAM a Mask R-CNN

Esta distinción refuerza la utilidad de Grad-CAM como herramienta de apoyo en el análisis interpretativo de arquitecturas de segmentación por instancias.

Además, una de las principales aportaciones prácticas de esta técnica en contextos de segmentación es su utilidad como indicador de confianza. Cuando el mapa de atención generado se solapa espacialmente con la máscara segmentada, esto puede interpretarse como una señal de consistencia entre el proceso interno del modelo y su salida final. Por el contrario, cuando se observa una desconexión significativa entre ambas (por ejemplo, cuando la atención se focaliza en regiones irrelevantes o periféricas), esto puede anticipar una segmentación deficiente. Tal como se analiza en [Figura 10.9](#) y [Figura 10.10](#), dicha desalineación coincide con un fallo evidente en la predicción, lo que sugiere que Grad-CAM puede utilizarse como herramienta complementaria para detectar casos de baja fiabilidad en modelos de segmentación. Esta aplicación refuerza su valor no solo como técnica explicativa, sino también como posible criterio para filtrar o revisar resultados críticos.

En este trabajo se ha implementado una adaptación funcional de Grad-CAM a Mask R-CNN, siguiendo la metodología descrita en los trabajos de Inbaraj et al. [[XJ₂₁](#), [XVM⁺₂₁](#), [XVM⁺₂₂](#)]. Como ya se ha comentado, estos autores han demostrado que es posible integrar Grad-CAM en arquitecturas de segmentación de instancias sin modificar la estructura del modelo, utilizando capas intermedias como `roi_align_classifier` y `mrcnn_class_logits` para obtener mapas de atención asociados a una clase y región específicas. Esta adaptación permite visualizar las regiones internas de la red que han contribuido más significativamente a una predicción, incluso en presencia de múltiples objetos, oclusión o ruido.

A partir de esta base teórica, se desarrolló un modelo auxiliar en TensorFlow/Keras que extrae las activaciones relevantes y calcula los gradientes necesarios para construir el mapa Grad-CAM para una RoI concreta. Esta implementación, se describe detalladamente a continuación:

1. Se selecciona la región de interés que se desea analizar, identificada mediante su índice dentro del conjunto de detecciones producidas por el modelo.
2. La imagen procesada (*molded image*), los metadatos de la imagen (*image meta*) y las anclas generadas durante la inferencia se preparan en el formato requerido por el modelo.
3. Se calcula el gradiente del logit correspondiente a la clase predicha para la RoI seleccionada con respecto a las activaciones obtenidas tras la capa `roi_align_classifier`, utilizando `GradientTape` de TensorFlow.
4. Se realiza un promedio global de estos gradientes a lo largo de los ejes espaciales para obtener los pesos α_k que cuantifican la importancia de cada canal de activación.
5. Finalmente, se genera el mapa de calor Grad-CAM como una combinación lineal ponderada de los mapas de características, seguido de una normalización y aplicación de ReLU para resaltar únicamente las regiones con mayor influencia positiva sobre la predicción.

Esta adaptación permite interpretar de manera localizada la decisión del modelo, proporcionando un mapa de atención para cada instancia detectada. Es especialmente útil en

8. Métodos Empleados

aplicaciones médicas como la segmentación dental, donde resulta esencial comprender qué regiones de la imagen han motivado la clasificación y segmentación de un diente concreto. Cabe destacar que el enfoque descrito mantiene la estructura de Mask R-CNN sin necesidad de modificar sus pesos ni realizar reentrenamiento.

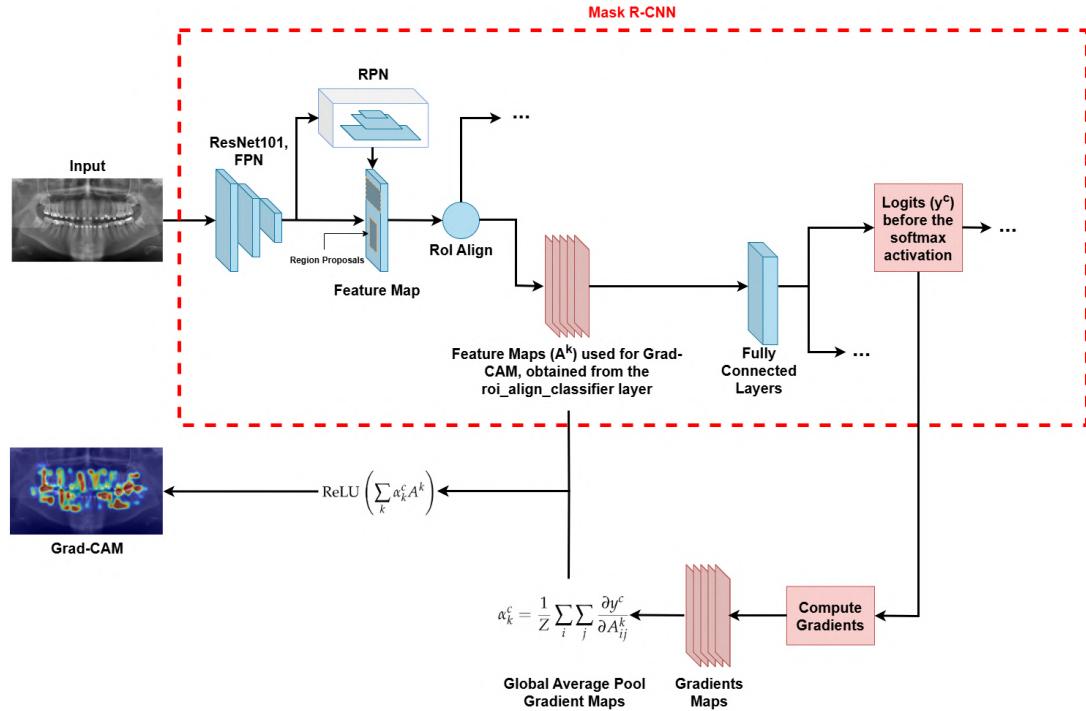


Figura 8.9.: Esquema del proceso de adaptación de Grad-CAM a la arquitectura Mask R-CNN, implementado en el presente trabajo. A partir de los mapas de activación generados por la capa `roi_align_classifier`, y de los logits de clase (y^c) obtenidos antes del softmax, se calculan los gradientes por retropropagación. Estos se promedian espacialmente para obtener los pesos α_k^c , que ponderan la contribución de cada canal. El mapa Grad-CAM resultante representa las regiones internas que más influyeron en la predicción.

9. Implementación

En este capítulo se describe la implementación práctica del trabajo desarrollado, incluyendo tanto el diseño del software como el entorno de ejecución utilizado. Se detallan los repositorios creados, la estructura del código, los principales módulos del proyecto y los pasos seguidos para el entrenamiento, evaluación y aplicación de técnicas de explicabilidad al modelo Mask R-CNN. Asimismo, se documentan las especificaciones del equipo empleado y las dificultades técnicas que surgieron durante el desarrollo experimental.

9.1. Diseño del software

El diseño del software no se ha orientado a crear una aplicación funcional para ser empleada por un usuario final, sino que su propósito principal es permitir el entrenamiento del modelo Mask R-CNN, la evaluación de su rendimiento y la aplicación de técnicas de explicabilidad mediante Grad-CAM.

Para la gestión del código y documentación, se ha utilizado el sistema de control de versiones *Git*, junto con la plataforma *GitHub*. En total, se han creado dos repositorios:

- **Repositorio TFG:** Contiene la planificación y elaboración de la memoria del trabajo de fin de grado. El enlace a dicho repositorio es el siguiente: <https://github.com/juanmaarg6/TFG>.
- **Repositorio DentCAM_RCNN:** Contiene el conjunto de datos utilizado en los experimentos, así como todos los archivos necesarios para la instanciación, entrenamiento y evaluación del modelo Mask R-CNN (se ha trabajado sobre una implementación preexistente compatible con TensorFlow 2.14.0, basada en el [repositorio de z-mahmud22](#), el cual a su vez es una versión adaptada del [repositorio de matterport](#), siendo este último compatible con TensorFlow 1.X). El enlace a dicho repositorio es el siguiente: https://github.com/juanmaarg6/DentCAM_RCNN.

Dentro del repositorio DentCAM_RCNN, se encuentra un notebook de Jupyter, el cual engloba todas las fases del proceso experimental, llamado `dentcam_rcnn.ipynb`. En líneas generales, se realizan los siguientes pasos:

- **Carga del conjunto de datos:** Lectura de las imágenes y máscaras de segmentación para su uso en el modelo.
- **Instanciación del modelo Mask R-CNN:** Configuración de los hiperparámetros modelo.
- **Fine-tuning del modelo:** Ajuste de los pesos preentrenados en COCO mediante entrenamiento en nuestro conjunto de datos específico.
- **Evaluación del rendimiento:** Cálculo de métricas para medir la calidad de la segmentación y la precisión en la detección de objetos.

9. Implementación

- **Aplicación de Grad-CAM:** Implementación de la técnica de explicabilidad Grad-CAM para visualizar las regiones del modelo que contribuyen a la detección de las *bounding boxes*.

Cabe destacar que, como ya se comentó en la [Sección 8.3](#), la implementación de Grad-CAM en este proyecto no se ha limitado a aplicar una solución ya existente, sino que ha requerido el desarrollo de una adaptación funcional específica para la arquitectura Mask R-CNN. Dado que esta red opera sobre múltiples regiones de interés (RoIs), se diseñó un modelo auxiliar capaz de acceder simultáneamente a los mapas de activación generados por la capa `roi_align_classifier` y a los logits de clasificación de la capa `mrcnn_class_logits`. Esta adaptación permite aplicar Grad-CAM de forma localizada sobre cada instancia detectada de manera individual, lo que resulta especialmente relevante en tareas de segmentación por instancias, donde es fundamental comprender las decisiones del modelo a nivel de objeto. Mediante la API `GradientTape` de TensorFlow, se calcularon los gradientes del logit correspondiente a la clase predicha para una RoI concreta, y se promediaron espacialmente para obtener la importancia relativa de cada canal de activación. Con esta información, se generó un mapa de atención localizado, que fue posteriormente reescalado al tamaño de la detección y proyectado sobre la imagen original. El resultado final se visualizó como un mapa de calor superpuesto a la imagen, permitiendo analizar visualmente qué regiones han influido en la decisión del modelo. Esta adaptación técnica se encuentra completamente integrada en el notebook `dentcam_rcnn.ipynb`, concretamente en la función `compute_gradcam(...)`, y constituye una de las aportaciones clave de este trabajo en el ámbito de la explicabilidad.

A continuación se presenta la estructura de paquetes del proyecto de Mask R-CNN, tal como se ilustra en la [Figura 9.1](#):

- **config:** Contiene la clase principal `Config` para gestionar todos los parámetros de entrenamiento, como tamaño de lote, número de pasos, tasa de aprendizaje, entre otros. Cualquier nueva configuración específica de este proyecto se derivaría de esta clase.
- **model:** Incluye la implementación base de la arquitectura Mask R-CNN. Es uno de los núcleos del framework donde se define la lógica de entrenamiento y predicción.
- **visualize:** Reúne funciones para la representación gráfica de resultados, permitiendo mostrar las máscaras de segmentación, *bounding boxes* y puntuaciones de confianza de forma clara y ordenada.
- **parallel_model:** Implementa la lógica necesaria para entrenar el modelo en paralelo o en múltiples GPUs, optimizando el proceso de entrenamiento cuando se cuenta con hardware adicional.
- **utils:** Contiene utilidades generales y funciones auxiliares (como transformaciones de imágenes, manejo de archivos, entre otras) que se utilizan de forma transversal en los distintos módulos.

En [Figura 9.2](#), [Figura 9.3](#), [Figura 9.4](#) y [Figura 9.5](#), presentamos también los diagramas de clases de los distintos paquetes del proyecto de Mask R-CNN (cabe destacar que el paquete `visualize` no contiene ninguna clase, por lo que no puede realizarse un diagrama de clases). Cada figura ilustra la estructura interna de las clases, mostrando atributos y métodos relevantes, así como las relaciones entre ellas.

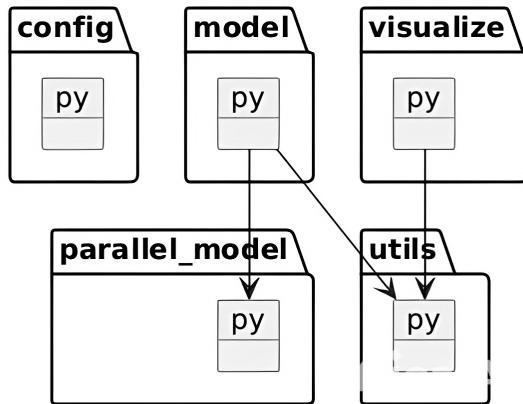


Figura 9.1.: Diagrama de paquetes del proyecto de Mask R-CNN.

Finalmente, en la [Figura 9.6](#) se muestra el diagrama de secuencia correspondiente a la ejecución del software desarrollado. Cabe destacar que el software diseñado no tiene como fin su uso por parte de terceros. De ser así, se habría creado una interfaz de usuario amigable en la cual, de manera intuitiva, se pudieran elegir las distintas opciones de ejecución.

9.2. Entorno de ejecución

Las ejecuciones se han realizado en un ordenador portátil ASUS TUF Gaming F17 FX707VI-HX040, con las siguientes especificaciones técnicas:

- **Sistema Operativo:** Windows 11 Home 64 bits.
- **Procesador:** Intel® Core™ i7-13620H (10 núcleos, 6 de alto rendimiento y 4 de eficiencia) con una frecuencia turbo de hasta 4.9 GHz.
- **Memoria RAM:** 32 GB DDR5 a 4800 MHz en configuración de doble canal.
- **Tarjeta gráfica:** NVIDIA GeForce RTX 4070 con 8 GB de memoria GDDR6.

Para la ejecución experimental, se ha creado un entorno de desarrollo en Anaconda con Python 3.10.12, garantizando la compatibilidad del software y la correcta gestión de dependencias. A continuación, se detallan las versiones de los paquetes principales utilizados en el desarrollo, junto con una breve descripción de su función:

- **Numpy:** 1.23.5 – Biblioteca para el manejo eficiente de arreglos y operaciones matemáticas en Python.
- **Pandas:** 0.23.3 – Biblioteca para la manipulación y análisis de datos estructurados en forma de tablas y series temporales.
- **Matplotlib:** 3.7.1 – Biblioteca para la generación de gráficos y visualización de datos.
- **TensorFlow (con soporte CUDA):** 2.14.0 – Framework de aprendizaje profundo optimizado para la aceleración en GPU.

9. Implementación

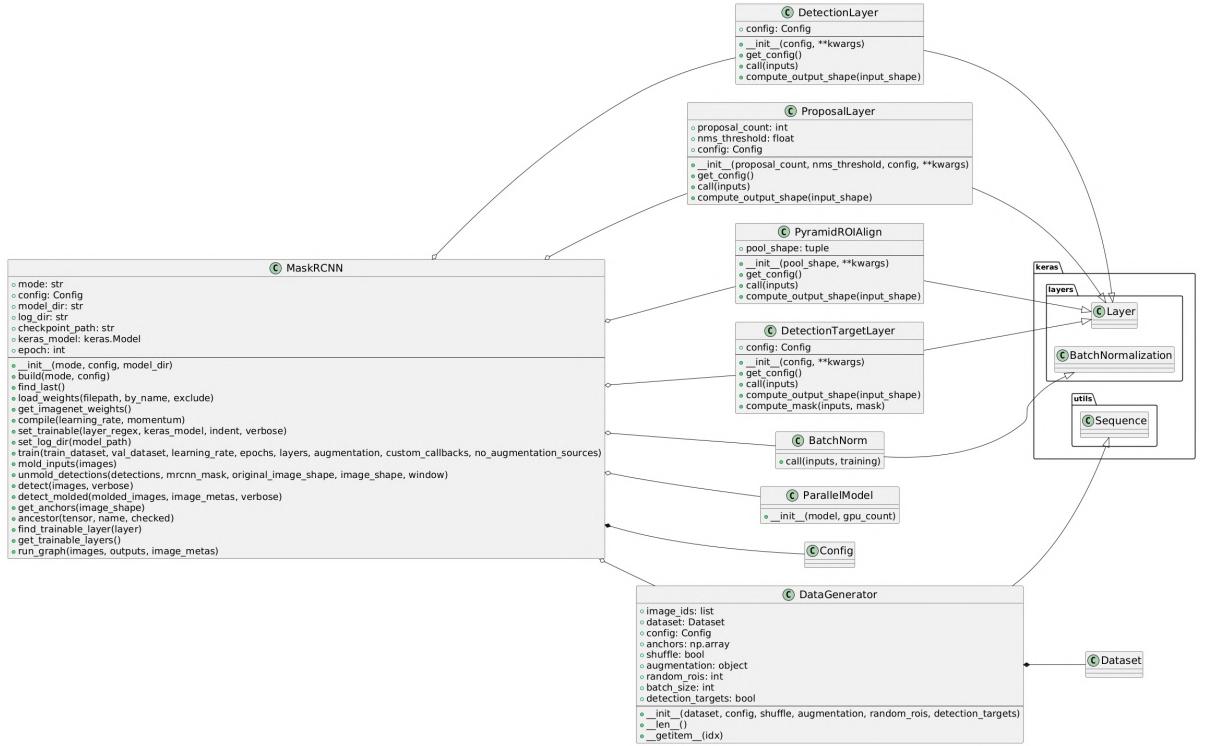


Figura 9.2.: Diagrama de clases del paquete model de Mask R-CNN.

- **OpenCV:** 4.8.0.76 – Biblioteca de visión por computadora para el procesamiento de imágenes y videos.
- **Keras:** 2.14.0 – API de alto nivel para la creación y entrenamiento de redes neuronales en TensorFlow.
- **Scikit-Image:** 0.19.3 – Biblioteca para el procesamiento y análisis avanzado de imágenes.
- **SciPy:** 1.11.3 – Conjunto de herramientas científicas para cálculos matemáticos avanzados, incluyendo optimización y procesamiento de señales.
- **Scikit-learn:** 1.6.1 – Biblioteca para el aprendizaje automático, ofreciendo herramientas para clasificación, regresión, clustering y validación de modelos.

Cabe destacar que, a lo largo del desarrollo experimental, se presentaron diversas dificultades técnicas que condicionaron el progreso del trabajo. Inicialmente, se intentó realizar la ejecución del modelo Mask R-CNN en la plataforma *Google Colab*, aprovechando los recursos en la nube debido a que mi antiguo equipo no contaba con una GPU compatible para realizar tareas de entrenamiento, ni siquiera un proceso de ajuste fino de las capas superiores del modelo. Aunque en un principio la ejecución funcionaba correctamente, un cambio repentino en la versión de Python de Colab (de la 3.10 a la 3.11) provocó incompatibilidades con las

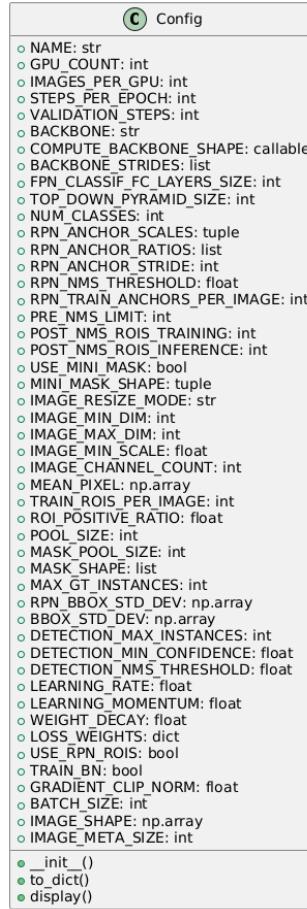


Figura 9.3.: Diagrama de clases del paquete config de Mask R-CNN.

dependencias necesarias para el proyecto, impidiendo la instalación de los paquetes requeridos mediante pip.

A esta situación se sumó un problema de hardware: mi antiguo equipo dejó de funcionar, lo que obligó a la adquisición de uno nuevo. En este contexto, se priorizó la compra de un dispositivo con una GPU de alto rendimiento que permitiera continuar con los experimentos en entorno local. El equipo finalmente adquirido, un ASUS TUF Gaming F17 FX707VI-HX040, permitió retomar el trabajo y llevar a cabo el ajuste fino de las capas superiores del modelo. Aún así, debido al elevado tamaño del modelo Mask R-CNN completo, no es posible entrenarlo desde cero en este nuevo equipo, por lo que se optó por reutilizar pesos preentrenados y adaptar las capas superiores del modelo al conjunto de datos específico del proyecto.

9. Implementación

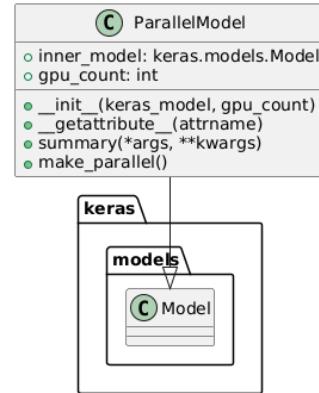


Figura 9.4.: Diagrama de clases del paquete parallel_model de Mask R-CNN.



Figura 9.5.: Diagrama de clases del paquete utils de Mask R-CNN.

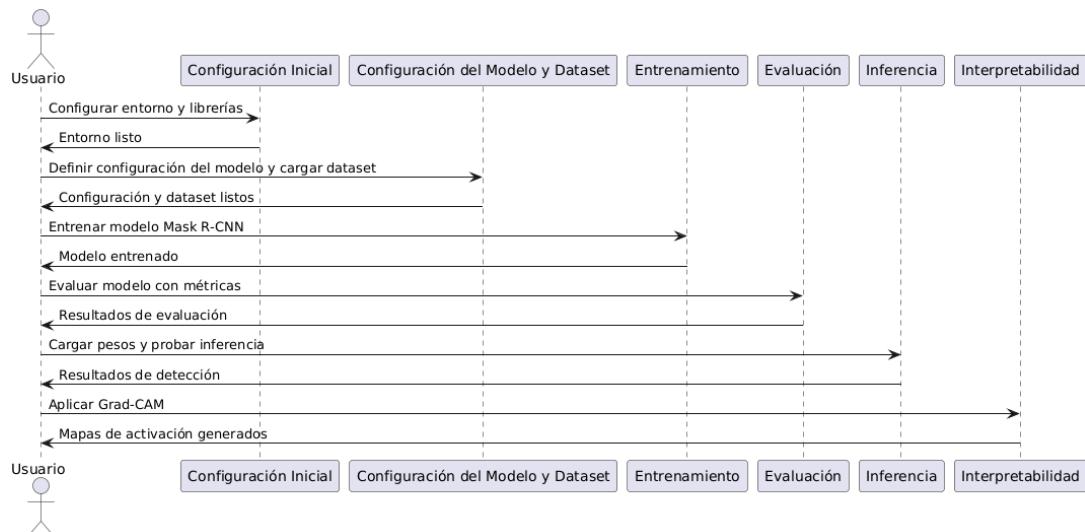


Figura 9.6.: Diagrama de secuencia del software empleado.

10. Experimentación

En este capítulo se detalla el proceso experimental llevado a cabo para el desarrollo de este trabajo. Se presentan los datos empleados y las métricas seleccionadas para evaluar el rendimiento de los modelos. Además, se describen los experimentos realizados, los resultados obtenidos y su análisis crítico. Este enfoque estructurado permite garantizar la reproducibilidad de los experimentos y ofrece una base sólida para interpretar los hallazgos presentados.

10.1. Datos empleados

El conjunto de datos empleado en este trabajo es el DNS (Detection, Numbering, and Segmentation) Panoramic Images, el cual surge a partir de la evolución y mejora de dos conjuntos de datos previos ampliamente utilizados en el análisis de imágenes dentales: el UFBA-UESC Dental Images y el UFBA-UESC Dental Images Deep. Estos conjuntos de datos representan hitos importantes en el desarrollo de herramientas basadas en aprendizaje profundo para la segmentación y numeración dental en radiografías panorámicas.

El UFBA-UESC Dental Images fue introducido por Silva et al. [SOP18] como un conjunto de datos grande y diverso, compuesto por 1500 imágenes panorámicas, categorizadas en diez grupos según características como restauraciones dentales, aparatos, implantes y dientes faltantes. Este conjunto de datos fue creado por la colaboración entre la Universidade Federal da Bahia (UFBA) y la Universidade Estadual de Santa Cruz (UESC), dos instituciones brasileñas que contribuyeron significativamente a la investigación en análisis de imágenes dentales. Diseñado inicialmente para tareas de segmentación semántica, este conjunto trataba los dientes como un único objeto, utilizando máscaras binarias para separarlos del resto de la imagen. Esto permitió desarrollar métodos de segmentación básicos que identificaban el área de los dientes, pero no diferenciaban dientes individuales ni proporcionaban información detallada para aplicaciones avanzadas. En sus experimentos, Silva et al. concluyeron que las soluciones basadas en aprendizaje profundo, como Mask R-CNN, superaban con creces los métodos clásicos basados en regiones, umbrales o contornos. Este conjunto de datos fue clave para demostrar, por primera vez, las ventajas del aprendizaje profundo en radiografías panorámicas, que presentan desafíos adicionales debido a la complejidad anatómica y la inclusión de estructuras óseas circundantes. Sin embargo, a pesar de sus contribuciones, el conjunto carecía de información detallada para segmentación por instancias o numeración dental, lo que limitaba su utilidad en tareas más complejas.

Para superar estas limitaciones, Jader et al. [JFR⁺18] adaptaron el UFBA-UESC Dental Images para incluir información de segmentación por instancias y crearon un nuevo conjunto denominado UFBA-UESC Dental Images Deep. Este proceso consistió en separar manualmente los dientes individuales a partir de las máscaras binarias que representaban el arco dental completo. Como resultado, el conjunto anotado incluía información de instancias individuales, facilitando tareas de segmentación más precisas. El UFBA-UESC Dental Images Deep fue

10. Experimentación

utilizado para investigar la detección y segmentación de dientes en radiografías panorámicas, aunque en este caso no incluía información de numeración dental. Este avance permitió analizar los dientes como instancias individuales, proporcionando un recurso valioso para investigaciones más detalladas en segmentación dental.

Finalmente, el conjunto DNS Panoramic Images, introducido por Silva et al. [SPOP20], surge como una evolución del UFBA-UESC Dental Images Deep, con anotaciones adicionales que incluyen la numeración dental según la notación FDI (Federación Dental Internacional). Véase [Figura 10.1](#). Este nuevo conjunto de datos mantiene las imágenes panorámicas originales, pero se enfoca en tareas más complejas como la detección, segmentación por instancias y numeración de dientes. De las 1500 imágenes originales, se anotaron 543 imágenes con información de numeración, mientras que las 778 imágenes restantes se emplearon para tareas de segmentación semántica. El DNS Panoramic Images se caracteriza por su alta diversidad, incluyendo imágenes con variaciones significativas en calidad, contraste y características anatómicas. Algunas categorías incluyen dientes restaurados, supernumerarios, faltantes o con dispositivos protésicos. Las imágenes de categorías que incluían dientes deciduos e implantes (categorías 5 y 6) fueron descartadas para garantizar consistencia en los resultados. El DNS Panoramic Images ha sido diseñado específicamente para abordar los desafíos de las radiografías panorámicas, ofreciendo un recurso clave para el desarrollo y evaluación de modelos avanzados de aprendizaje profundo.

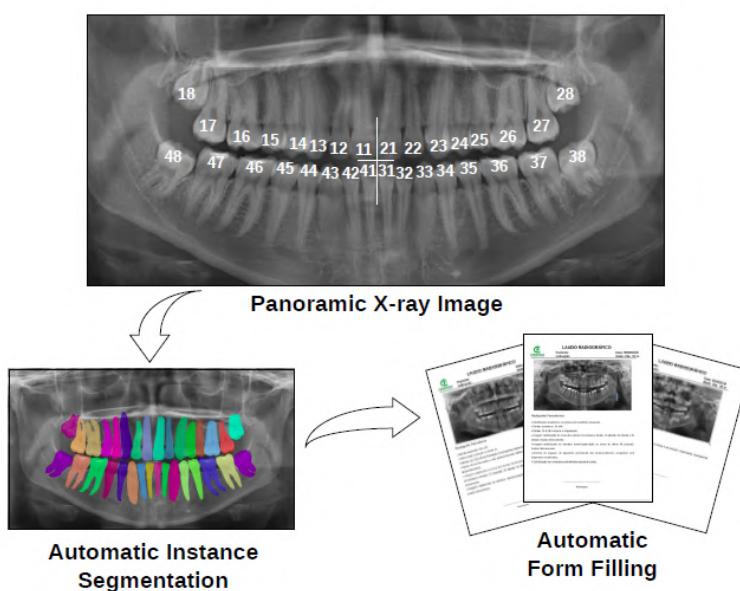


Figura 10.1.: La notación de dos dígitos de la FDI es utilizada por los dentistas para registrar sus hallazgos, donde el número, la forma y la ubicación de los dientes son información esencial para apoyar la toma de decisiones en odontología. Imagen extraída de [SPOP20].

En este trabajo, utilizamos las 543 imágenes anotadas con información de numeración del

10.1. Datos empleados

conjunto de datos DNS Panoramic Images, cuyas características se detallan en la [Tabla 10.1](#). De este total, 376 imágenes se utilizaron para entrenamiento (un 85 % aproximadamente), 87 imágenes para validación (un 15 % aproximadamente) y 80 imágenes para test (un 15 % aproximadamente), distribuyendo de igual forma las imágenes de cada categoría, como se resume en la [Tabla 10.2](#).

Categoría	Descripción	Número de imágenes
1	Imágenes con todos los dientes, que incluyen dientes restaurados y con aparatos dentales.	23
2	Imágenes con todos los dientes, que incluyen dientes restaurados y sin aparatos dentales.	174
3	Imágenes con todos los dientes, sin restauraciones y con aparatos dentales.	42
4	Imágenes con todos los dientes, sin restauraciones y sin aparatos dentales.	92
5	Imágenes que contienen implantes dentales.	-
6	Imágenes que contienen más de 32 dientes.	-
7	Imágenes con dientes faltantes, que incluyen dientes restaurados y con aparatos dentales.	36
8	Imágenes con dientes faltantes, que incluyen dientes restaurados y sin aparatos dentales.	128
9	Imágenes con dientes faltantes, sin restauraciones y con aparatos dentales.	14
10	Imágenes con dientes faltantes, sin restauraciones y sin aparatos dentales.	34
		543

Tabla 10.1.: Características del conjunto de datos DNS Panoramic Images usado en este trabajo.

En [Figura 10.2](#) se muestran imágenes representativas de las distintas categorías del conjunto de datos DNS Panoramic Images, destacando la diversidad de condiciones dentales presentes, como dientes completos, faltantes, etc. Para cada radiografía dental, se dispone además de una máscara correspondiente que identifica y segmenta individualmente cada diente. Un ejemplo de estas máscaras se presenta en la [Figura 10.3](#), donde se resalta la estructura detallada de los dientes en una radiografía panorámica perteneciente a la categoría 1.

En la [Figura 10.4](#) se presentan ejemplos adicionales del conjunto de datos empleado, formados por pares de imágenes compuestos por la radiografía panorámica original y su correspondiente máscara de segmentación. Estas parejas representan la entrada que recibe el modelo para el entrenamiento, donde cada diente es identificado como una instancia separada mediante distintas tonalidades.

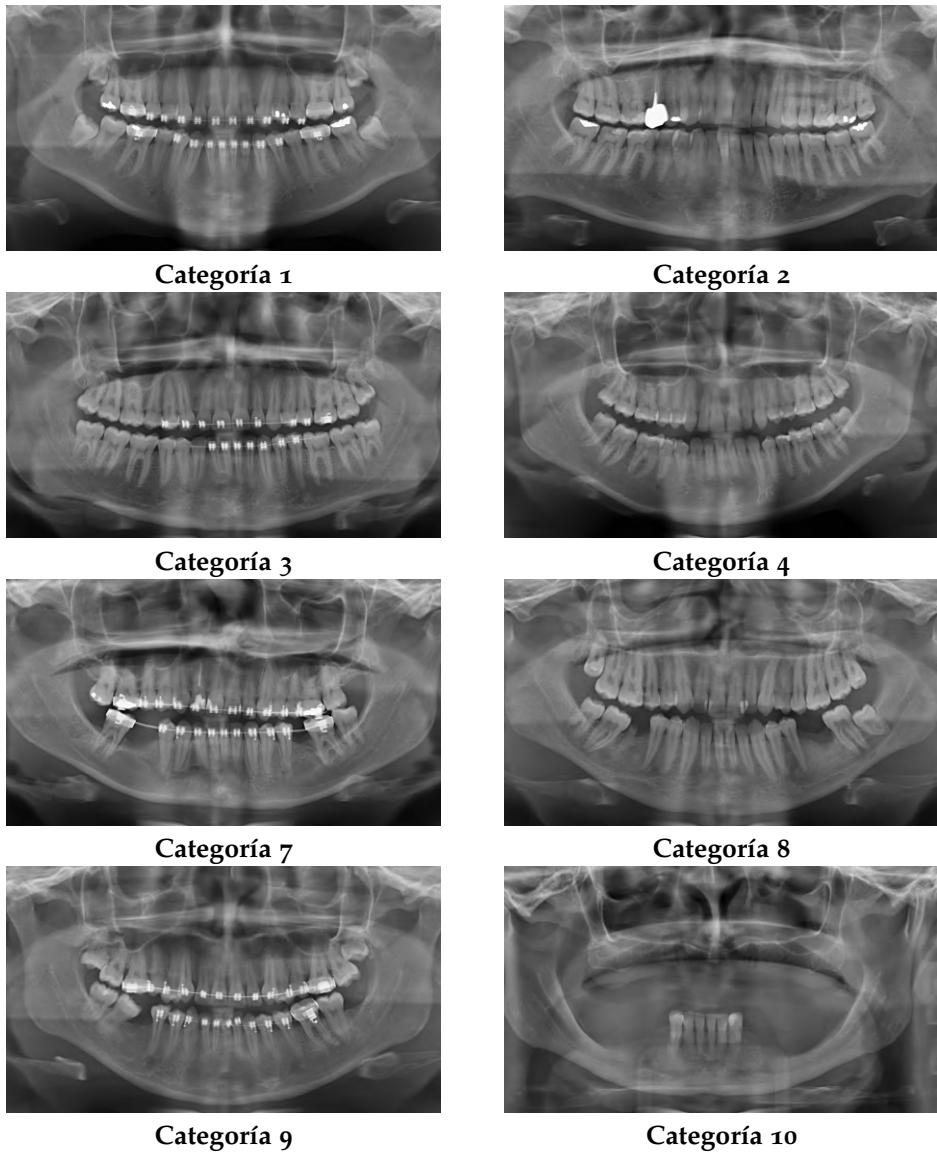


Figura 10.2.: Ejemplos representativos de radiografías dentales panorámicas clasificadas según las categorías del conjunto de datos DNS Panoramic Images.

Categoría	Total	Entrenamiento	Validación	Test
1	23	16	4	3
2	175	122	27	26
3	42	29	7	6
4	93	65	14	14
7	35	24	6	5
8	127	88	20	19
9	14	9	3	2
10	34	23	6	5
	543	376	87	80

Tabla 10.2.: Distribución del número de imágenes del conjunto de datos DNS Panoramic Images en los subconjuntos de entrenamiento, validación y test, según las categorías definidas.



Figura 10.3.: Ejemplo de máscara dental generada para segmentación de instancias, donde cada tono de gris identifica un diente individual. La imagen muestra la estructura separada de cada pieza dental en una radiografía panorámica perteneciente a la categoría 1.

10. Experimentación

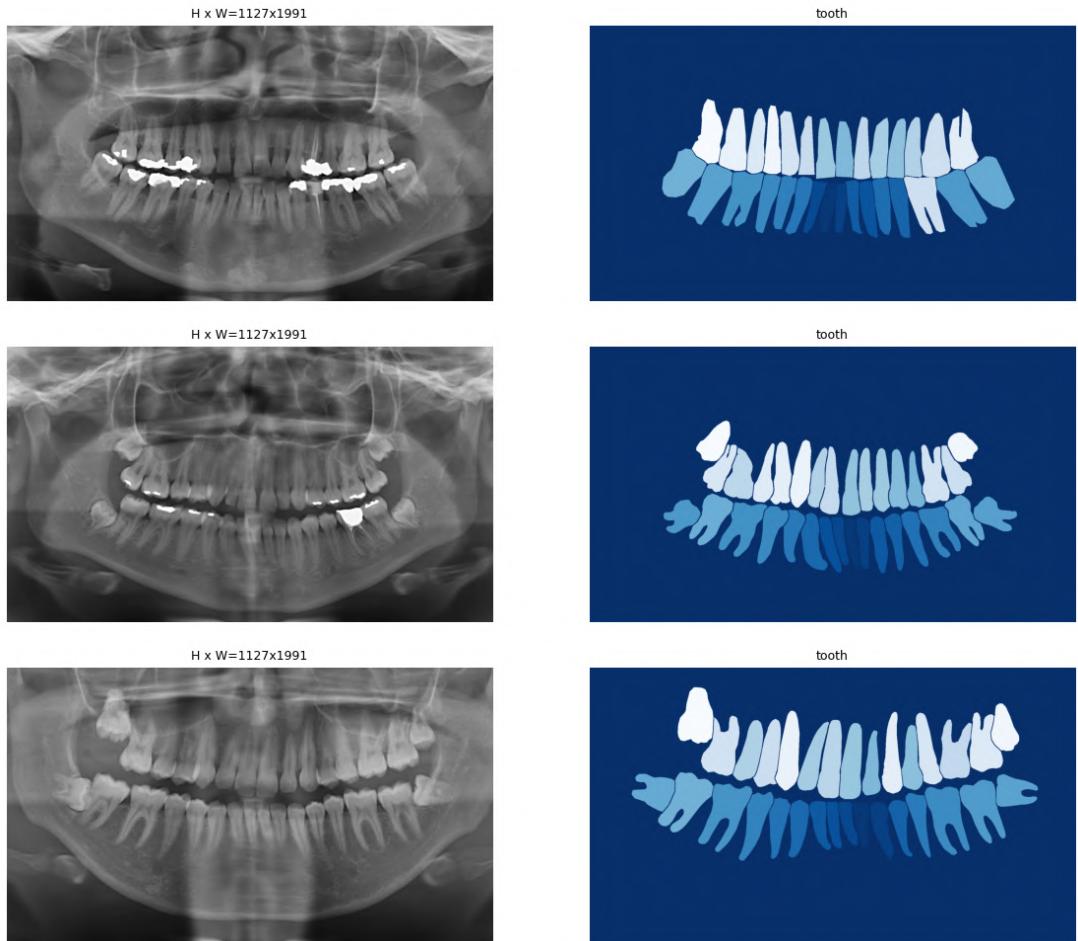


Figura 10.4.: Ejemplos de radiografías dentales panorámicas junto con sus respectivas máscaras de segmentación. Las imágenes fueron seleccionadas aleatoriamente del conjunto de entrenamiento del conjunto de datos y muestran, a la izquierda, la radiografía original y, a la derecha, la segmentación correspondiente donde los diferentes tonos de azul representan distintas piezas dentales.

10.2. Configuración de hiperparámetros

La configuración de hiperparámetros representa una fase crucial en el proceso de entrenamiento de redes neuronales profundas, ya que determina en gran medida la eficiencia, estabilidad y capacidad de generalización del modelo. A diferencia de los parámetros internos, que se ajustan automáticamente mediante algoritmos de optimización como el descenso por gradiente, los hiperparámetros deben definirse manualmente antes del inicio del entrenamiento. En este trabajo, se han seleccionado cuidadosamente los valores más adecuados para entrenar el modelo Mask R-CNN sobre el conjunto de datos dentales.

Entre los hiperparámetros más destacados se encuentra la tasa de aprendizaje fijada en 0.001, un valor comúnmente empleado en fases iniciales que permite un aprendizaje estable sin comprometer la convergencia. El tamaño del batch se estableció en 2, teniendo en cuenta las limitaciones de memoria propias del entrenamiento con imágenes de alta resolución. Además, se definieron 200 pasos por época y un total de 10 pasos de validación, configuraciones que permiten un control regular sobre la evolución del aprendizaje. Otro hiperparámetro clave es la arquitectura del *backbone*, que en este caso corresponde a resnet101, una red profunda con buenos resultados en tareas de extracción de características complejas.

10.3. Métricas empleadas para la evaluación

10.3.1. Métricas de detección

10.3.1.1. Precision, Recall y F1-Score

Para evaluar el rendimiento del modelo en la tarea de segmentación de instancias dentales, se utilizan métricas clásicas de la evaluación en aprendizaje automático: *Precision*, *Recall* y *F1-Score*. Estas métricas se calculan a partir de la comparación entre las predicciones del modelo y las anotaciones reales (*ground truth*) de cada imagen, permitiendo cuantificar tanto la exactitud como la cobertura del modelo en la detección de objetos.

Los conceptos clave que intervienen en su definición son los siguientes:

- ***True Positives (TP)***: Número de dientes correctamente detectados por el modelo.
- ***False Positives (FP)***: Número de detecciones incorrectas, es decir, regiones predichas por el modelo que no se corresponden con ningún diente real.
- ***False Negatives (FN)***: Número de dientes reales que el modelo no ha sido capaz de detectar.

A partir de estos valores, se definen las métricas de la siguiente manera:

$$\text{Precision} = \frac{TP}{TP + FP},$$

$$\text{Recall} = \frac{TP}{TP + FN},$$

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.$$

10. Experimentación

La *Precision* mide la proporción de predicciones correctas sobre el total de predicciones realizadas por el modelo. En otras palabras, indica cuántas de las detecciones realizadas corresponden realmente a dientes.

Por otra parte, la *Recall* evalúa la capacidad del modelo para identificar todos los dientes presentes en una imagen, midiendo la proporción de dientes correctamente detectados respecto al total de dientes reales.

Finalmente, el *F1-Score* representa la media armónica entre *Precision* y *Recall*, ofreciendo una medida equilibrada del rendimiento cuando existe un compromiso entre ambas métricas.

10.3.1.2. Average Precision

El *Average Precision* (AP) es una de las métricas más utilizadas en tareas de detección y segmentación de instancias para evaluar el rendimiento del modelo. Su valor se corresponde con el área bajo la curva de *Precision-Recall* (PR) para una categoría determinada, lo que proporciona una medida global de la precisión del modelo en función de distintos niveles de exhaustividad (*Recall*).

El cálculo del AP se basa en la siguiente integral:

$$AP = \int_0^1 P(R)dR,$$

donde $P(R)$ es la función de *Precision* en función del *Recall*. En la práctica, esta integral se aproxima numéricamente a partir de puntos discretos obtenidos durante el proceso de evaluación.

Existen variantes del AP que se definen en función del umbral de solapamiento (IoU, por sus siglas en inglés) requerido para considerar una predicción como correcta. En particular, en este trabajo se ha empleado la métrica AP@50, que calcula el AP con un umbral de IoU fijo del 50 %. Este valor implica que una predicción se considera correcta si el área de intersección entre la máscara predicha y la real representa al menos el 50 % del área de su unión.

10.3.2. Métricas de segmentación

10.3.2.1. Dice Similarity Coefficient

El *Dice Similarity Coefficient* (DSC) es una medida de similitud entre dos conjuntos binarios: la máscara predicha y la máscara real. Su valor oscila entre 0 (sin solapamiento) y 1 (solapamiento perfecto), y se define como:

$$DSC = \frac{2|A \cap B|}{|A| + |B|}$$

donde A representa la máscara predicha y B la máscara real. Esta métrica penaliza tanto los falsos positivos como los falsos negativos y es especialmente útil cuando las clases están desbalanceadas.

10.3.2.2. Jaccard Index (Intersection over Union)

El *Jaccard Index*, también conocido como *Intersection over Union* (IoU), mide la proporción entre el área de intersección y el área de unión de las máscaras predicha y real. Se expresa como:

$$IoU = \frac{|A \cap B|}{|A \cup B|}$$

El valor del IoU también varía entre 0 y 1, siendo 1 el solapamiento perfecto. A diferencia del *Dice Similarity Coefficient*, el IoU penaliza de forma más severa los errores cuando el solapamiento entre las máscaras es bajo.

10.3.2.3. Hausdorff Distance

La *Hausdorff Distance* (HD) mide la máxima distancia entre los bordes de dos conjuntos, proporcionando una estimación del peor caso en la disimilitud entre contornos. Formalmente, se define como:

$$HD(A, B) = \max \left\{ \sup_{a \in A} \inf_{b \in B} d(a, b), \sup_{b \in B} \inf_{a \in A} d(b, a) \right\}$$

donde $d(a, b)$ representa la distancia euclídea entre los píxeles a y b . Esta métrica es particularmente útil para evaluar el grado de alineación entre los bordes de las máscaras predichas y las reales, siendo sensible a errores localizados, incluso si el resto de la máscara presenta buen solapamiento.

10.3.3. Métricas de explicabilidad

10.3.3.1. Deletion e Insertion

Como se definió en la Subsección 6.5.3, la fidelidad es una propiedad clave de los métodos de explicabilidad, ya que indica si las regiones señaladas por una técnica como relevantes para la predicción reflejan de forma precisa el comportamiento real del modelo. Para cuantificar esta propiedad, se emplean métricas como *Deletion* e *Insertion*, que permiten evaluar de manera empírica si eliminar o introducir dichas regiones afecta significativamente a la predicción.

Estas métricas no miden directamente la fidelidad, pero actúan como indicadores cuantitativos de ella: si al eliminar los píxeles más relevantes según el mapa Grad-CAM la probabilidad de la clase objetivo cae rápidamente, o si al introducirlos sobre una imagen en negro la probabilidad crece con rapidez, se considera que el mapa posee una alta fidelidad.

La métrica *Deletion* [FV17] mide cómo decrece la probabilidad de la clase objetivo a medida que se van eliminando (enmascarando) las regiones más relevantes según el mapa Grad-CAM. Cuanto más rápido decrece, mayor es la fidelidad del mapa. Formalmente, se define como el área bajo la curva:

$$\text{Deletion} = \int_0^1 p_c(t) dt,$$

10. Experimentación

donde $p_c(t)$ es la probabilidad de la clase c cuando se ha eliminado una fracción t de los píxeles más relevantes.

La métrica *Insertion* [PDS18] mide cómo aumenta la probabilidad de la clase objetivo al ir introduciendo progresivamente las regiones más relevantes sobre una imagen base (por ejemplo, una imagen completamente en negro). Cuanto más rápido aumenta la probabilidad, más informativo es el mapa. También se calcula como el área bajo la curva:

$$\text{Insertion} = \int_0^1 p_c(t) dt,$$

donde en este caso $p_c(t)$ representa la probabilidad de la clase c al insertar un porcentaje t de los píxeles más relevantes.

Ambas métricas están acotadas en el rango $[0, 1]$, ya que se definen como áreas bajo curvas normalizadas. En el caso de *Insertion*, valores cercanos a 1 indican que las regiones destacadas por Grad-CAM son altamente informativas, ya que su incorporación incrementa rápidamente la confianza del modelo en la clase predicha. Por el contrario, en *Deletion*, valores bajos (cercanos a 0) son deseables, ya que reflejan una fuerte caída en la probabilidad de la clase al eliminar los píxeles más relevantes. Por tanto, un mapa Grad-CAM con alta fidelidad tenderá a mostrar un AUC alto en *Insertion* y bajo en *Deletion*, lo que permite comparar objetivamente la calidad explicativa de distintos modelos o técnicas.

10.3.3.2. Stability

Como se expuso en la Subsección 6.5.3, la estabilidad es una propiedad fundamental de los métodos de explicabilidad, ya que refleja hasta qué punto las explicaciones generadas por el modelo son robustas frente a pequeñas perturbaciones en los datos de entrada. Una técnica estable debe proporcionar mapas consistentes para entradas prácticamente equivalentes, evitando variaciones arbitrarias que dificulten su interpretación.

Para evaluar cuantitativamente esta propiedad, se utiliza una métrica de similitud entre mapas, basada en la correlación de Pearson [ZBPG22]. Esta métrica no mide la estabilidad de manera directa, pero actúa como un indicador cuantitativo: si al introducir una perturbación leve en la imagen, el mapa Grad-CAM resultante se mantiene muy similar al original, se considera que la técnica posee una alta estabilidad.

Formalmente, se define como:

$$\text{Stability}(x) = \rho(M_x, M_{x+\delta}),$$

donde M_x y $M_{x+\delta}$ son los mapas Grad-CAM generados a partir de la imagen original x y su versión perturbada $x + \delta$, respectivamente, y ρ representa el coeficiente de correlación de Pearson.

Esta métrica toma valores entre -1 y 1 , donde valores cercanos a 1 indican una alta consistencia entre mapas, y por tanto, mayor estabilidad de la explicación frente a variaciones mínimas en la entrada. Valores próximos a 0 reflejan una baja correlación, lo que sugiere que la explicación es sensible a pequeños cambios y, por tanto, menos fiable. En el caso extremo,

valores negativos implicarían una correlación inversa entre los mapas, lo que sería indicativo de inestabilidad severa. Para garantizar que la comparación sea válida, se verifica previamente que la clase predicha por el modelo no se haya visto alterada por la perturbación, es decir, que $f(x) = f(x + \delta)$. En contextos prácticos, se consideran deseables valores altos de correlación (por encima de 0.7) para asegurar una explicación robusta y reproducible.

10.3.3.3. Jaccard Index entre Grad-CAM y máscara de segmentación

Aunque Grad-CAM no genera una máscara binaria como tal, es posible cuantificar la coherencia espacial entre el mapa de atención generado y la máscara de segmentación predicha por el modelo mediante el *Jaccard Index* (IoU). Esta métrica permite evaluar si el modelo ha focalizado su atención en las mismas regiones sobre las que ha aplicado la segmentación, proporcionando así una medida de consistencia entre el proceso de toma de decisiones interno y la salida generada.

Para ello, el mapa Grad-CAM se binariza aplicando un umbral relativo (por ejemplo, el percentil 90), generando así un conjunto A de píxeles relevantes. La máscara segmentada por el modelo constituye el conjunto B . El índice de Jaccard entre ambos se calcula como:

$$IoU_{\text{Grad-CAM}} = \frac{|A \cap B|}{|A \cup B|},$$

donde $|A \cap B|$ representa el número de píxeles comunes entre el mapa de atención binarizado y la máscara de segmentación predicha, mientras que $|A \cup B|$ corresponde al número total de píxeles únicos presentes en ambos.

Esta métrica toma valores en el rango $[0, 1]$, donde 0 indica una ausencia total de solapamiento entre atención y segmentación, y 1 implica una coincidencia perfecta. En el contexto de explicabilidad, valores elevados de $IoU_{\text{Grad-CAM}}$ indican que el modelo ha dirigido su atención hacia las regiones que ha terminado segmentando, lo que sugiere una mayor coherencia interna en el proceso decisional. Por el contrario, valores bajos pueden evidenciar que el modelo ha tomado decisiones basadas en regiones poco informativas o incluso espurias, comprometiendo la interpretabilidad del resultado.

Si bien el $IoU_{\text{Grad-CAM}}$ no forma parte del conjunto tradicional de métricas de explicabilidad, su inclusión en este trabajo se justifica como una medida de fidelidad estructural, complementaria a métricas funcionales como *Deletion* e *Insertion*. Esta métrica ha sido propuesta en el presente estudio con el objetivo de analizar la consistencia espacial entre la atención del modelo y las regiones segmentadas en arquitecturas como Mask R-CNN, permitiendo así una valoración adicional de la calidad explicativa de los mapas Grad-CAM.

Esta métrica responde directamente a uno de los objetivos planteados en la introducción de este trabajo: evaluar hasta qué punto la atención interna del modelo, estimada por Grad-CAM, se alinea espacialmente con las regiones segmentadas por Mask R-CNN. De este modo, el $IoU_{\text{Grad-CAM}}$ permite materializar numéricamente esa coherencia o consistencia entre atención y segmentación que se planteó como criterio indirecto de confiabilidad. Su análisis cuantitativo en las siguientes secciones permitirá valorar en qué medida dicha consistencia es alta en casos de segmentación correcta, y si disminuye en ejemplos fallidos o dudosos, aportando así una perspectiva adicional sobre la robustez del modelo.

10.4. Hipótesis iniciales

La experimentación desarrollada en este capítulo parte de las siguientes hipótesis principales:

1. Un modelo Mask R-CNN preentrenado con pesos generales no es capaz de segmentar correctamente las piezas dentales en radiografías panorámicas, debido a la ausencia de conocimiento específico del dominio odontológico.
2. Mediante un proceso de ajuste fino sobre un conjunto de datos especializado, es posible adaptar el modelo al contexto dental y mejorar significativamente su rendimiento.
3. El modelo ajustado puede llegar a superar tanto a métodos clásicos de segmentación como a otras arquitecturas más complejas, siempre que se realice una correcta adaptación al dominio.

Los experimentos desarrollados y evaluados a continuación tienen como objetivo validar estas hipótesis, evaluando el rendimiento del modelo base, el impacto del ajuste fino y su comparación con enfoques alternativos.

10.5. Evaluación y análisis de resultados

10.5.1. Comparación entre modelo base y modelo ajustado

En este trabajo se ha empleado la arquitectura Mask R-CNN para abordar la tarea de segmentación de piezas dentales en radiografías panorámicas. Se parte de un modelo con pesos preentrenados en el conjunto de datos COCO (*Common Objects in Context*) [LMB⁺14], un referente en Visión por Computador que incluye más de 300.000 imágenes anotadas para tareas de detección y segmentación, aunque no contempla clases relacionadas con el ámbito dental.

A pesar de esta limitación, el preentrenamiento en COCO proporciona una base sólida para la extracción de características visuales generales. Sobre esta base, se ha llevado a cabo un proceso de ajuste fino utilizando exclusivamente imágenes del conjunto DNS Panoramic Images, con el objetivo de adaptar el modelo al contexto específico de la anatomía dental.

Durante este proceso, se ha mantenido la arquitectura original del modelo y se han optimizado únicamente los parámetros relevantes para la detección y segmentación en el nuevo dominio, debido a restricciones computacionales que impidieron reentrenar toda la red. Esta personalización permite al modelo mejorar sustancialmente su precisión en la identificación de dientes individuales como instancias separadas, lo cual se refleja tanto en los resultados cuantitativos como cualitativos presentados más adelante.

Comenzamos analizando los resultados cuantitativos obtenidos por el modelo Mask R-CNN en sus dos variantes: el modelo base, preentrenado en el conjunto COCO sin ajuste al dominio dental, y el modelo ajustado mediante ajuste fino con imágenes del conjunto DNS Panoramic Images.

La comparación detallada por categoría dental puede consultarse en la tabla [Tabla A.1](#) del Apéndice A para el modelo base y en la [Tabla B.1](#) del Apéndice B para el modelo ajustado.

La [Tabla 10.3](#) presentada a continuación resume las métricas promedio por partición. Las métricas consideradas incluyen Precision, Recall, F1-Score y AP@50 para la detección; y DSC, IoU y Hausdorff Distance (HD) para la segmentación.

Tal como era de esperar, el modelo base muestra un rendimiento muy limitado. En el conjunto de test, alcanza un F1-Score promedio de 0.0571, un AP@50 de apenas 0.0109, y valores de DSC e IoU de 0.0125 y 0.0097, respectivamente. Además, la HD se sitúa en 30.73 píxeles. Estos resultados reflejan una baja capacidad del modelo para detectar o segmentar correctamente piezas dentales, debido a que COCO no incluye dicha clase ni imágenes morfológicamente similares.

En contraste, el modelo ajustado presenta una mejora drástica en todas las métricas. En test, alcanza un F1-Score de 0.9604, un AP@50 de 0.9398, un DSC de 0.8776 y un IoU de 0.7923, mientras que la HD se reduce a 12.26 píxeles. Estos resultados confirman que el ajuste fino, incluso aplicado de forma parcial por limitaciones computacionales, ha sido suficiente para adaptar el modelo al dominio dental con un alto grado de precisión y consistencia.

Modelo	Partición	Precision	Recall	F1-Score	AP@50	DSC	IoU	HD
Base	Train	0.0583	0.0669	0.0626	0.0111	0.0143	0.0115	27.57
	Validation	0.0490	0.0603	0.0542	0.0137	0.0125	0.0099	27.89
	Test	0.0523	0.0628	0.0571	0.0109	0.0125	0.0097	30.73
Ajustado	Train	0.9869	0.9620	0.9741	0.9585	0.9017	0.8123	10.13
	Validation	0.9606	0.9528	0.9566	0.9479	0.8847	0.7952	11.55
	Test	0.9736	0.9480	0.9604	0.9398	0.8776	0.7923	12.26

Tabla 10.3.: Comparación de métricas promedio de detección y segmentación entre el modelo base y el modelo ajustado de Mask R-CNN. Se incluyen valores promedio para todas las categorías dentales. Se recuerda que las métricas Precision, Recall, F1-Score, AP@50, DSC e IoU se deben maximizar, mientras que la métrica HD debe minimizarse.

Esta evolución también se observa en los diagramas de caja ([Figura 10.5](#) y [Figura 10.6](#)) que muestran una gran dispersión y bajo rendimiento en el modelo base, frente a valores altos, estables y homogéneos en el modelo ajustado. En conjunto, los resultados cuantitativos evidencian la necesidad de ajustar los pesos del modelo para adaptarlo a contextos clínicos específicos, y demuestran la efectividad del proceso de ajuste fino para mejorar tanto la detección como la segmentación de estructuras anatómicas dentales.

Para complementar el análisis cuantitativo, se ha realizado una evaluación cualitativa del comportamiento del modelo Mask R-CNN en sus dos variantes (base y ajustado) con el fin de observar visualmente las diferencias en la segmentación de piezas dentales.

Se han seleccionado las imágenes con mejor rendimiento AP@50 por categoría en el conjunto de test, con el objetivo de analizar los casos más favorables de cada modelo ([Figura 10.9](#) y [Figura 10.10](#)).

Aun tratándose de sus mejores ejemplos, el modelo base genera máscaras incompletas, impre-

10. Experimentación

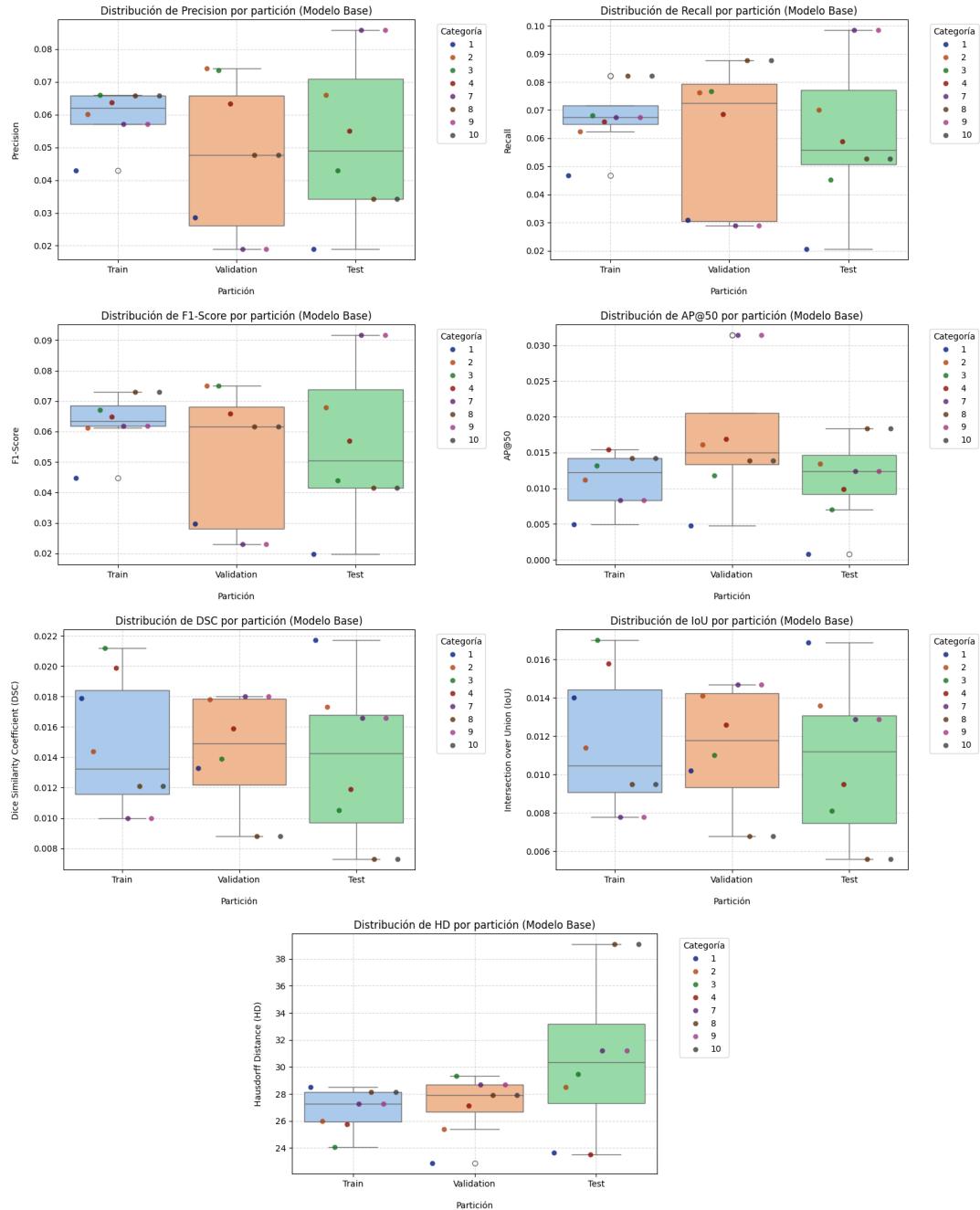


Figura 10.5.: Diagramas de caja (boxplots) que muestran la distribución de las métricas de evaluación por partición (entrenamiento, validación y test) para el modelo base de Mask R-CNN. Cada gráfico recoge la variación de una métrica específica (Precision, Recall, F1-Score, AP@50, DSC, IoU y HD) entre las distintas categorías dentales.

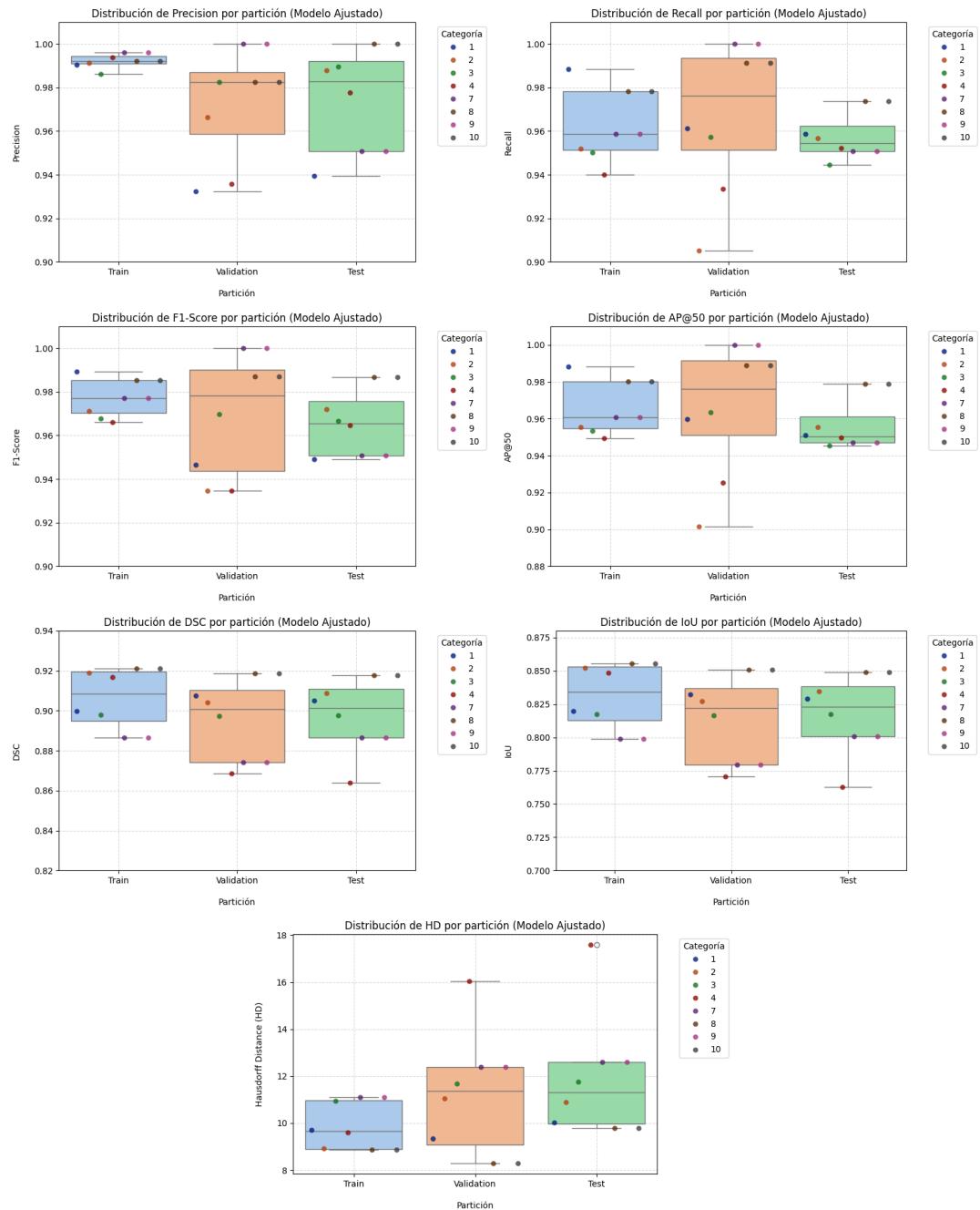


Figura 10.6.: Diagramas de caja (boxplots) que muestran la distribución de las métricas de evaluación por partición (entrenamiento, validación y test) para el modelo ajustado de Mask R-CNN. Cada gráfico recoge la variación de una métrica específica (Precision, Recall, F1-Score, AP@50, DSC, IoU y HD) entre las distintas categorías dentales.

10. Experimentación

cisas o inexistentes, y en ocasiones detecta regiones irrelevantes. Por el contrario, el modelo ajustado produce segmentaciones más coherentes y anatómicamente correctas, respetando la morfología y localización de las piezas dentales, con una drástica reducción de errores de sobresegmentación o falsas detecciones.

En conjunto, las observaciones visuales refuerzan las conclusiones cuantitativas: el modelo ajustado, adaptado al dominio dental mediante ajuste fino, presenta una mejora clara en la calidad de segmentación respecto al modelo base, tanto en términos de precisión como de consistencia visual.

Con el objetivo de examinar la capacidad explicativa de los modelos Mask R-CNN evaluados, se ha utilizado la técnica Grad-CAM para visualizar las regiones de la imagen que influyen más en sus predicciones. Este análisis permite comprobar si el modelo centra su atención en estructuras anatómicamente relevantes, como las piezas dentales, o si por el contrario actúa de forma arbitraria.

La comparación detallada por categoría dental puede consultarse en la tabla [Tabla A.2](#) del [Apéndice A](#) para el modelo base y en la [Tabla B.2](#) del [Apéndice B](#) para el modelo ajustado. La [Tabla 10.4](#) presentada a continuación resume las métricas promedio por partición. Las métricas consideradas incluyen Deletion, Insertion, Stability e $\text{IoU}_{\text{Grad-CAM}}$.

En el modelo base, preentrenado en el conjunto COCO sin ajuste al dominio dental, los resultados de Grad-CAM en el conjunto de test muestran un patrón problemático: valores extremadamente altos en Deletion (0.9938) e Insertion (0.9936), junto con una baja estabilidad (0.2472) y un $\text{IoU}_{\text{Grad-CAM}}$ muy reducido (0.1347). Estos resultados indican que el modelo no se apoya en regiones visualmente relevantes para sus decisiones, sino en patrones genéricos que no se corresponden con la anatomía dental. La eliminación de zonas destacadas no afecta a la predicción, lo que sugiere una baja fidelidad explicativa, y las activaciones suelen encontrarse fuera del área dental.

En cambio, el modelo ajustado, entrenado mediante ajuste fino sobre imágenes dentales, muestra un patrón opuesto en dicho conjunto de test: una caída pronunciada en Deletion (0.1595), Insertion elevada (0.9965), y mejoras notables en Stability (0.8891) e $\text{IoU}_{\text{Grad-CAM}}$ (0.4365). Estos valores reflejan que el modelo ha aprendido a centrar su atención en zonas pertinentes para la tarea de segmentación, generando mapas de activación más coherentes, reproducibles y alineados con las máscaras reales.

Esta evolución también se aprecia gráficamente en los diagramas de caja de la [Figura 10.7](#), que recogen la distribución de las métricas de explicabilidad por partición para el modelo base. Se observa una alta variabilidad y dispersión en todas las métricas, lo que refleja una atención poco consistente entre categorías y una escasa especialización del modelo. Este comportamiento contrasta con la mayor estabilidad observada en el modelo ajustado, cuyos resultados individuales se muestran en la figura [Figura 10.8](#), con valores más concentrados y centrados en rangos óptimos. En conjunto, estas visualizaciones refuerzan la conclusión de que el ajuste fino no solo mejora el rendimiento predictivo, sino también la coherencia y fiabilidad de las explicaciones generadas por Grad-CAM.

Esta diferencia cuantitativa se complementa con una inspección visual de los mapas genera-

Modelo	Partición	Deletion	Insertion	Stability	$\text{IoU}_{\text{Grad-CAM}}$
Base	Train	0.9929	0.9932	0.2564	0.1457
	Validation	0.9932	0.9933	0.2218	0.1316
	Test	0.9938	0.9936	0.2472	0.1347
Ajustado	Train	0.1612	0.9964	0.8806	0.4258
	Validation	0.1577	0.9963	0.8866	0.4214
	Test	0.1595	0.9965	0.8891	0.4365

Tabla 10.4.: Comparación de métricas promedio de explicabilidad entre el modelo base y el modelo ajustado de Mask R-CNN. Se presentan los promedios para cada partición, calculados con Grad-CAM. Se recuerda que las métricas Insertion, Stability e $\text{IoU}_{\text{Grad-CAM}}$ se deben maximizar, mientras que la métrica Deletion debe minimizarse.

dos. En los ejemplos más representativos de cada categoría se observa que el modelo base destaca regiones alejadas de los dientes, mientras que el modelo ajustado concentra su atención sobre las piezas dentales ([Figura 10.9](#) y [Figura 10.10](#)). Es importante señalar que esta revisión no constituye una evaluación cualitativa formal en el sentido estricto propuesto en la literatura de XAI, ya que no ha contado con la participación de un experto en el dominio odontológico. Las observaciones aquí descritas se basan en una interpretación técnica del alineamiento visual entre las activaciones de Grad-CAM y las regiones anatómicas dentales esperadas.

En conjunto, estos resultados evidencian que el modelo ajustado no solo mejora en rendimiento, sino también en coherencia explicativa: sus predicciones se apoyan en regiones anatómicamente relevantes y sus mapas de atención presentan mayor fidelidad, estabilidad y alineación con el objetivo clínico. Cabe destacar que, si bien podría esperarse que los mapas de Grad-CAM mostraran cierta correlación con las regiones segmentadas incluso en modelos poco precisos, los valores observados de $\text{IoU}_{\text{Grad-CAM}}$ en el modelo base desmienten esta suposición. Dicho modelo no solo presenta un rendimiento deficiente, sino que también genera mapas de atención poco informativos, lo que indica una falta de correspondencia entre sus activaciones internas y las estructuras anatómicas relevantes. Esta falta de alineación no debe interpretarse como una limitación de la técnica de explicabilidad, sino como un reflejo fiel del pobre razonamiento del modelo. Así, Grad-CAM actúa como una herramienta diagnóstica que permite evidenciar la desconexión entre la atención y la segmentación en modelos ineficaces, mientras que en modelos ajustados refuerza su fiabilidad al mostrar activaciones coherentes con las salidas generadas. Este análisis aporta un criterio adicional de evaluación que va más allá de las métricas tradicionales, permitiendo valorar la robustez y consistencia del proceso decisional interno. En este sentido, la métrica $\text{IoU}_{\text{Grad-CAM}}$ representa la materialización cuantitativa de la hipótesis formulada en la introducción de este trabajo: que una mayor coherencia espacial entre la atención interna del modelo y las regiones segmentadas puede interpretarse como un indicador de robustez. Su cálculo permite evaluar en qué medida el modelo está tomando decisiones basadas en características visuales pertinentes, y no en patrones espurios o irrelevantes. Así, se proporciona un doble criterio de validación (segmentación efectiva y atención coherente) que refuerza la confianza en los resultados generados por el modelo ajustado.

10. Experimentación

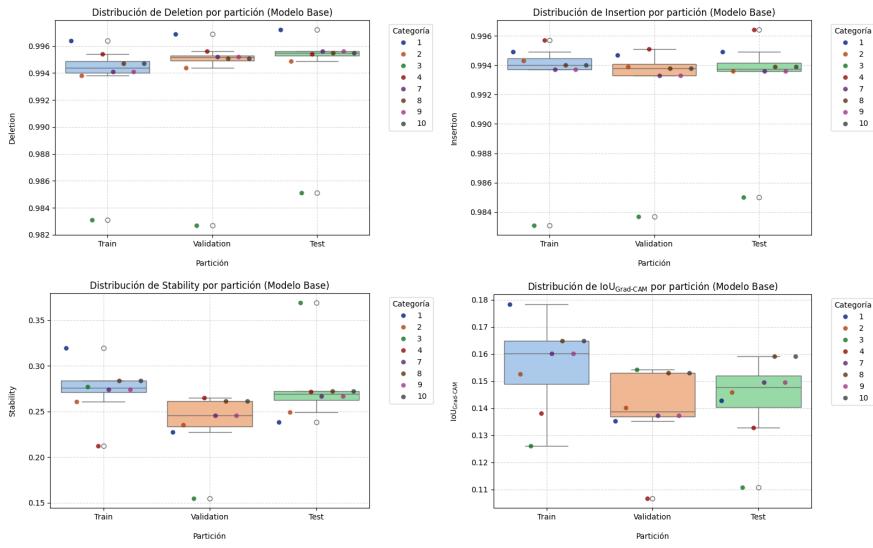


Figura 10.7.: Diagramas de caja (*boxplots*) que muestran la distribución de las métricas de explicabilidad por partición (entrenamiento, validación y test) para el modelo base de Mask R-CNN. Cada gráfico representa la variabilidad entre categorías dentales de una métrica concreta derivada de la aplicación de Grad-CAM (Deletion, Insertion, Stability e IoU_{Grad-CAM}).

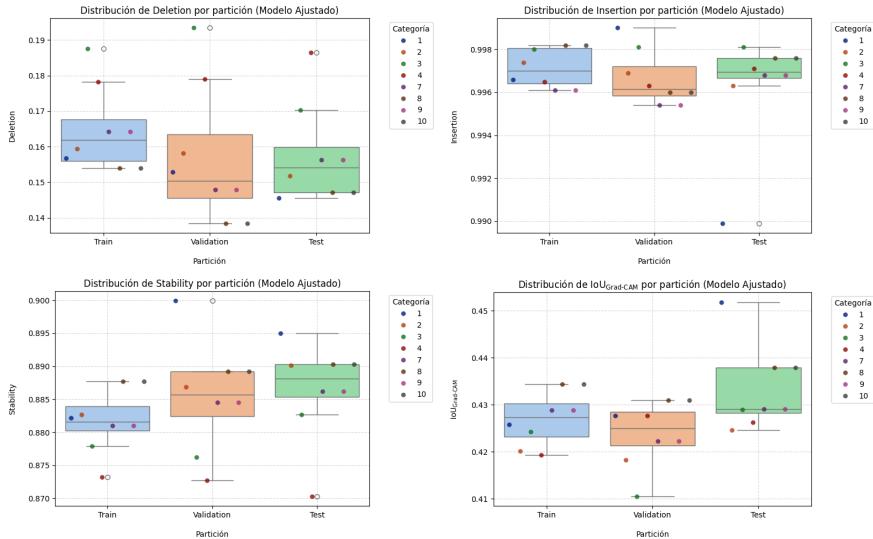


Figura 10.8.: Diagramas de caja (*boxplots*) que muestran la distribución de las métricas de explicabilidad por partición (entrenamiento, validación y test) para DentCAM-RCNN. Cada gráfico representa la variabilidad entre categorías dentales de una métrica concreta derivada de la aplicación de Grad-CAM (Deletion, Insertion, Stability e IoU_{Grad-CAM}).

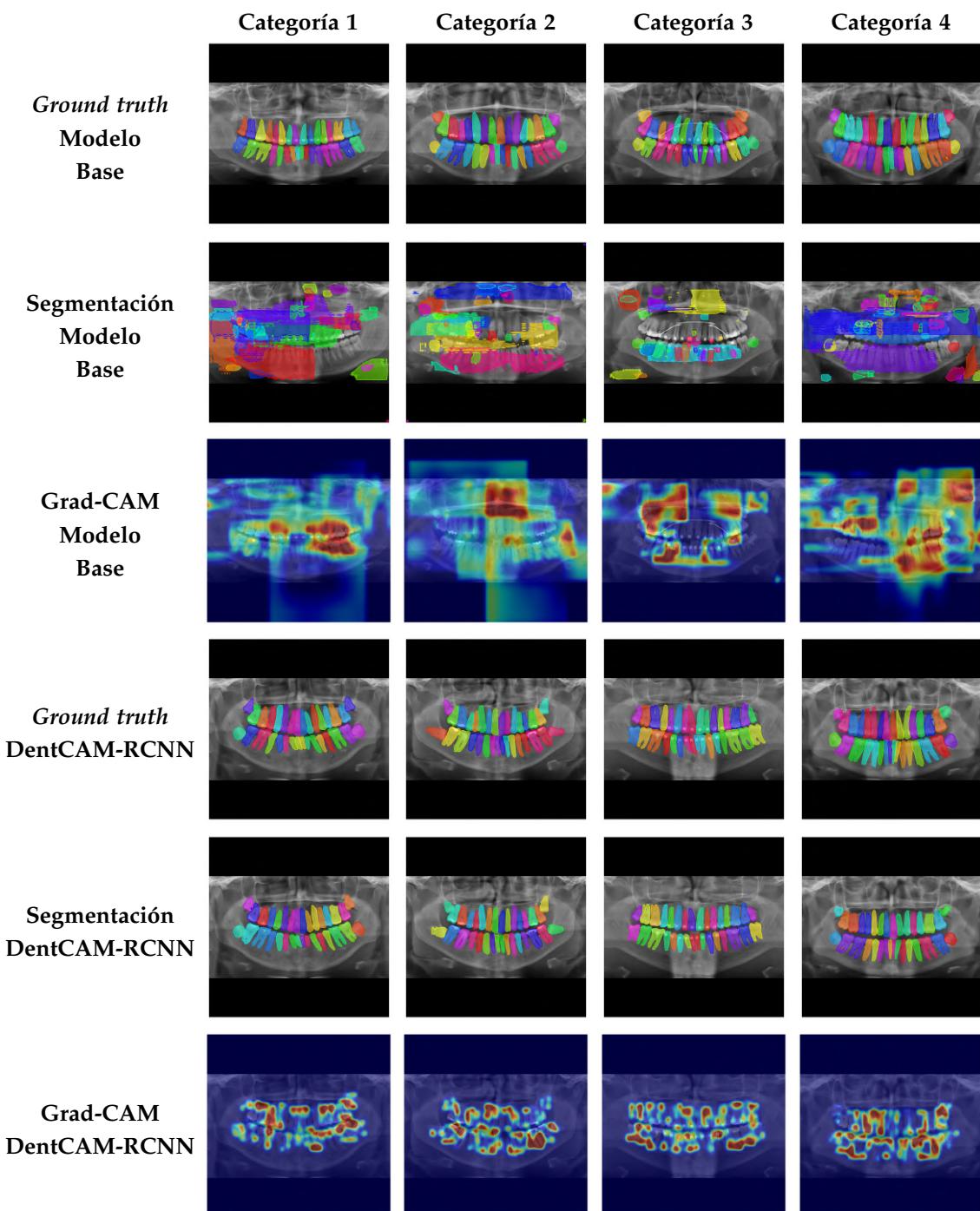


Figura 10.9.: Comparación visual entre la segmentación de referencia del modelo base y DentCAM-RCNN (*ground truth*), los resultados de segmentación y Grad-CAM de ambos modelos, en las imágenes con mejor rendimiento AP@50 para las categorías 1 a 4.

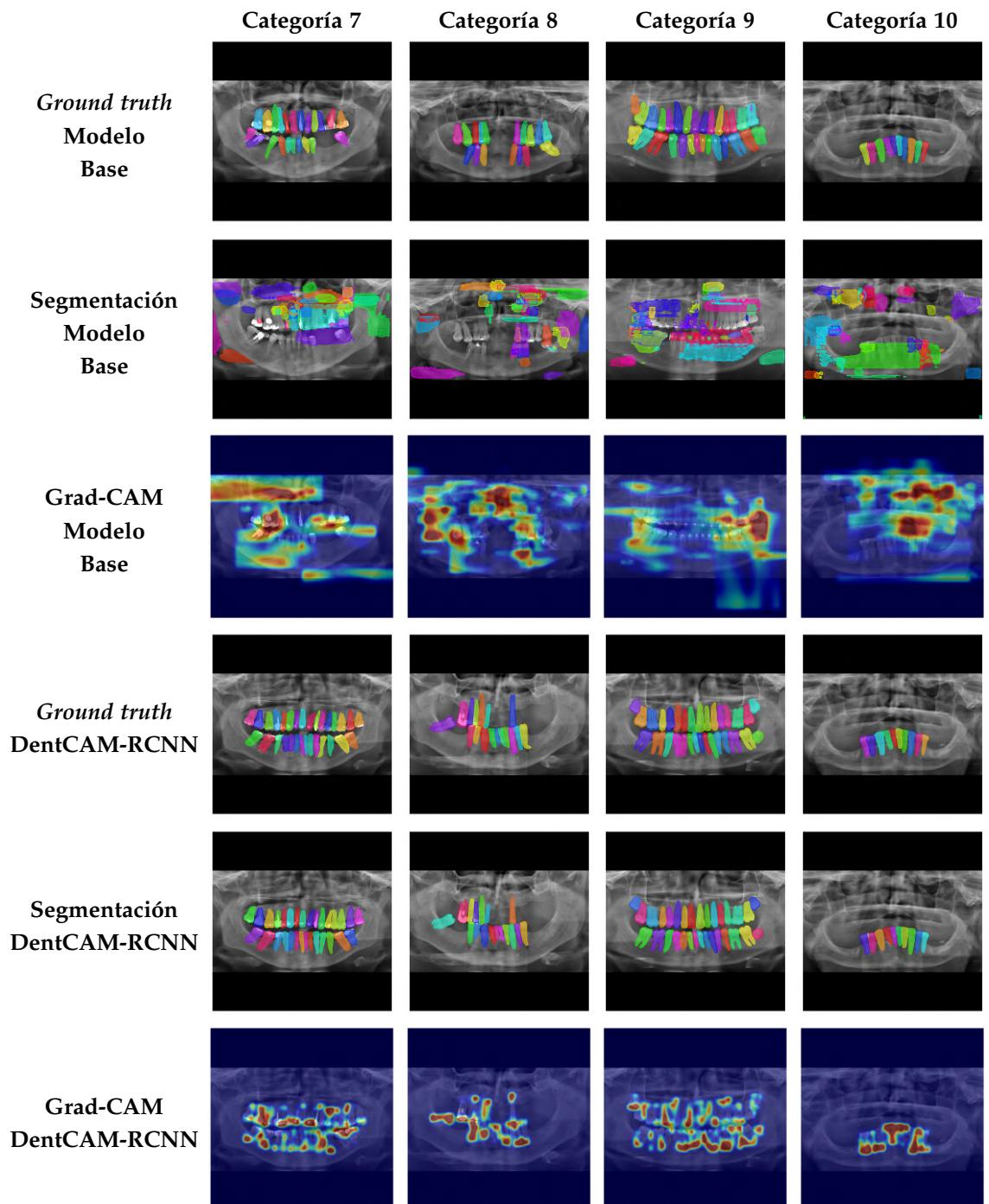


Figura 10.10.: Comparación visual entre la segmentación de referencia del modelo base y DentCAM-RCNN (*ground truth*), los resultados de segmentación y Grad-CAM de ambos modelos, en las imágenes con mejor rendimiento AP@50 para las categorías 7 a 10.

Esta propuesta desarrollada (basada en el ajuste fino del modelo Mask R-CNN al dominio dental y en la incorporación de la técnica de explicabilidad Grad-CAM) la denominamos **DentCAM-RCNN**. Este nombre sintetiza tanto el enfoque clínico específico, como la integración de capacidades explicativas y la arquitectura subyacente empleada. DentCAM-RCNN representa así una solución adaptada, precisa e interpretable para la segmentación automatizada de estructuras dentales en radiografías panorámicas.

10.5.2. Comparación de DentCAM-RCNN con métodos clásicos de segmentación

En la [Tabla 10.5](#) se comparan las métricas obtenidas por DentCAM-RCNN con una selección de métodos clásicos de segmentación no supervisados aplicados en el ámbito de la segmentación dental. Los resultados de estos métodos han sido recopilados a partir del estudio comparativo presentado en [\[JFR⁺18\]](#), que evalúa diversas aproximaciones tradicionales sobre imágenes dentales.

Como puede observarse, los métodos clásicos presentan un rendimiento muy variable dependiendo del enfoque utilizado. Por ejemplo, técnicas como el *global thresholding* o los métodos basados en *region growing* y *watershed* logran valores de F1-Score moderadamente aceptables (entre 0.44 y 0.61), aunque en general muestran una compensación entre precisión y exhaustividad que limita su efectividad clínica.

Otros métodos como *splitting/merging*, *Canny* o *Sobel* obtienen valores de F1-Score muy bajos, especialmente debido a una pobre capacidad de recuperación de regiones relevantes (bajo Recall), lo cual los hace poco adecuados para tareas exigentes como la segmentación precisa de dientes.

Frente a estos enfoques, DentCAM-RCNN destaca con una diferencia notable en todas las métricas. En el conjunto de test, alcanza una Precision del 97.76 %, un Recall del 94.73 % y un F1-Score del 96.22 %, superando ampliamente a todos los métodos tradicionales analizados.

Estos resultados ponen de manifiesto las limitaciones de los enfoques clásicos, que dependen en gran medida de heurísticas y parámetros específicos de cada imagen, y refuerzan el valor de los modelos de aprendizaje profundo ajustados al dominio para lograr una segmentación robusta, precisa y generalizable.

En definitiva, DentCAM-RCNN establece un nuevo estándar frente a las técnicas tradicionales en términos de precisión y aplicabilidad al análisis automatizado de radiografías dentales panorámicas.

Además del análisis cuantitativo, se ha llevado a cabo una comparativa cualitativa sobre múltiples categorías representativas, tal como se muestra en las [Figura 10.11](#) y [Figura 10.12](#). Las imágenes permiten apreciar visualmente las diferencias de segmentación entre los métodos clásicos y DentCAM-RCNN. Mientras que los métodos tradicionales tienden a generar contornos difusos, fragmentaciones o sobresegmentaciones, especialmente visibles en técnicas como *splitting/merging*, *Canny* o *Sobel*, el modelo de aprendizaje profundo logra contornos nítidos, continuos y anatómicamente coherentes en la mayoría de las categorías. Este comportamiento visual respalda los resultados métricos obtenidos y refuerza la capacidad del modelo para generalizar correctamente sobre distintas morfologías dentales, constituyéndose

Método	Precision	Recall	F1-Score
Region growing [MD11]	0.3500	0.6300	0.4400
Splitting/merging [SPA14]	0.8100	0.0800	0.1400
Global thresholding [HH08]	0.5200	0.6900	0.5600
Niblack method [RAZI14]	0.5100	0.8200	0.6100
Fuzzy C-means [Als15]	0.6100	0.4500	0.4900
Canny [RAH ⁺ 15]	0.4500	0.1100	0.1700
Sobel [RAH ⁺ 15]	0.6600	0.0300	0.0600
Active contours without edges [AEZ15]	0.5100	0.5700	0.5200
Level set method [RRN13]	0.4800	0.6800	0.5200
Watershed [LSSL12]	0.4800	0.8200	0.5800
DentCAM-RCNN	0.9776	0.9473	0.9622

Tabla 10.5.: Comparación de métricas de evaluación globales (todas las categorías) en la partición de test entre métodos de segmentación no supervisados extraídos de [JFR⁺18] y DentCAM-RCNN.

como una herramienta sólida y fiable para su uso clínico.

10.5.3. Comparación de DentCAM-RCNN con estudios recientes de segmentación de instancias dentales en 2D

En esta sección se presenta una comparación cuantitativa entre DentCAM-RCNN y otros enfoques recientes que emplean la arquitectura Mask R-CNN para la segmentación de instancias dentales en imágenes panorámicas bidimensionales. Todos los estudios considerados han sido analizados previamente en la Sección 7.1 de este trabajo, donde se detallan sus principales características, conjuntos de datos y métricas de evaluación.

Como se observa en la Tabla 10.6, el modelo desarrollado en este trabajo alcanza el F1-Score más alto (0.9604) entre los estudios comparados, y presenta un rendimiento competitivo en cuanto al Dice Similarity Coefficient (0.8776) y el IoU (0.7923). Si bien el estudio de Lee et al. [LHK⁺20] reporta un IoU ligeramente superior (0.8770), su F1-Score es menor y se basa en un conjunto de datos notablemente ampliado mediante técnicas de aumento. En contraste, nuestro enfoque logra resultados robustos sobre un volumen de datos intermedio, lo que evidencia una mejor capacidad de generalización. En conjunto, estos resultados consolidan la eficacia del modelo propuesto para la segmentación precisa y automatizada de estructuras dentales en radiografías panorámicas.

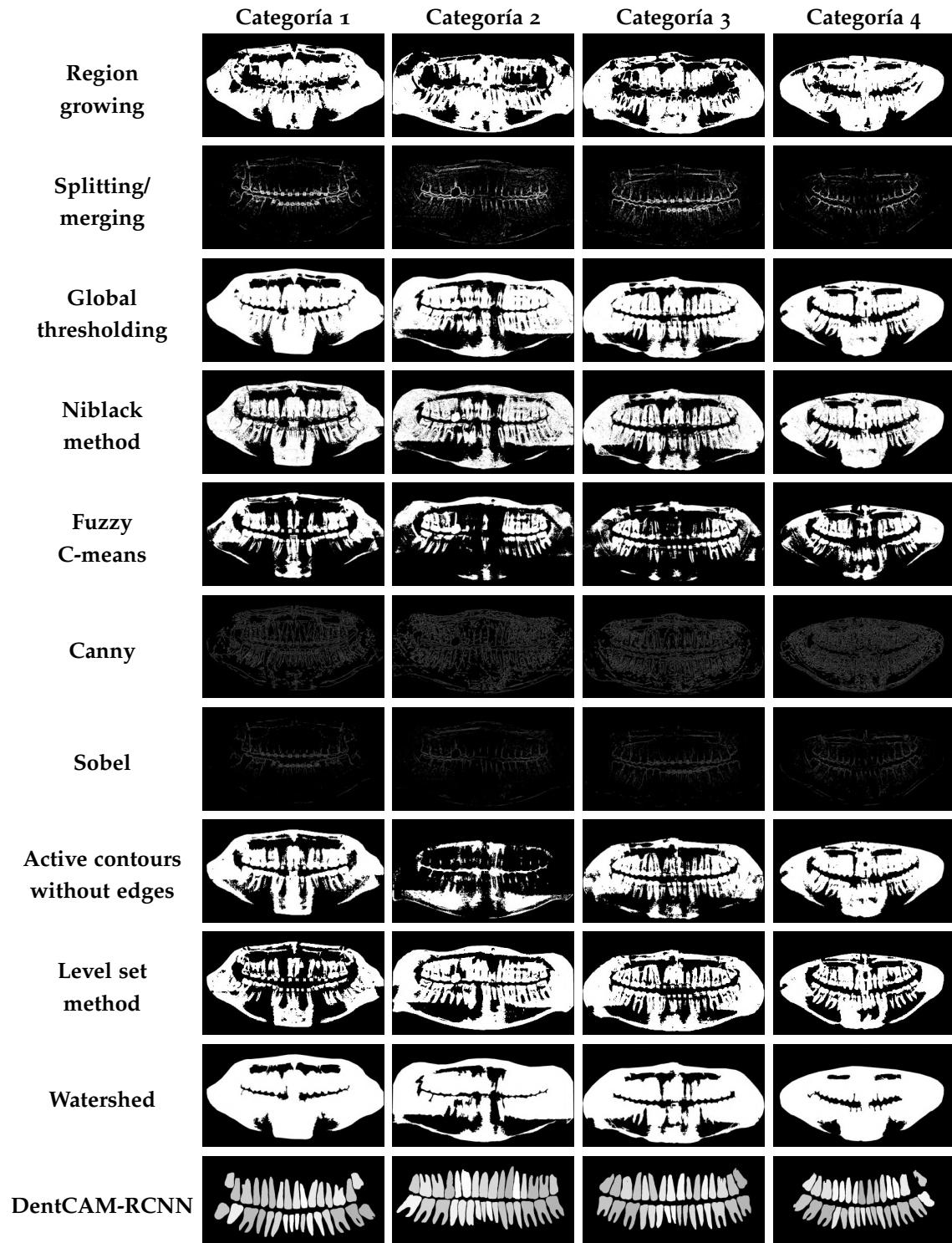


Figura 10.11.: Comparación cualitativa entre métodos clásicos de segmentación no supervisados y DentCAM-RCNN sobre las categorías 1 a 4.

10. Experimentación

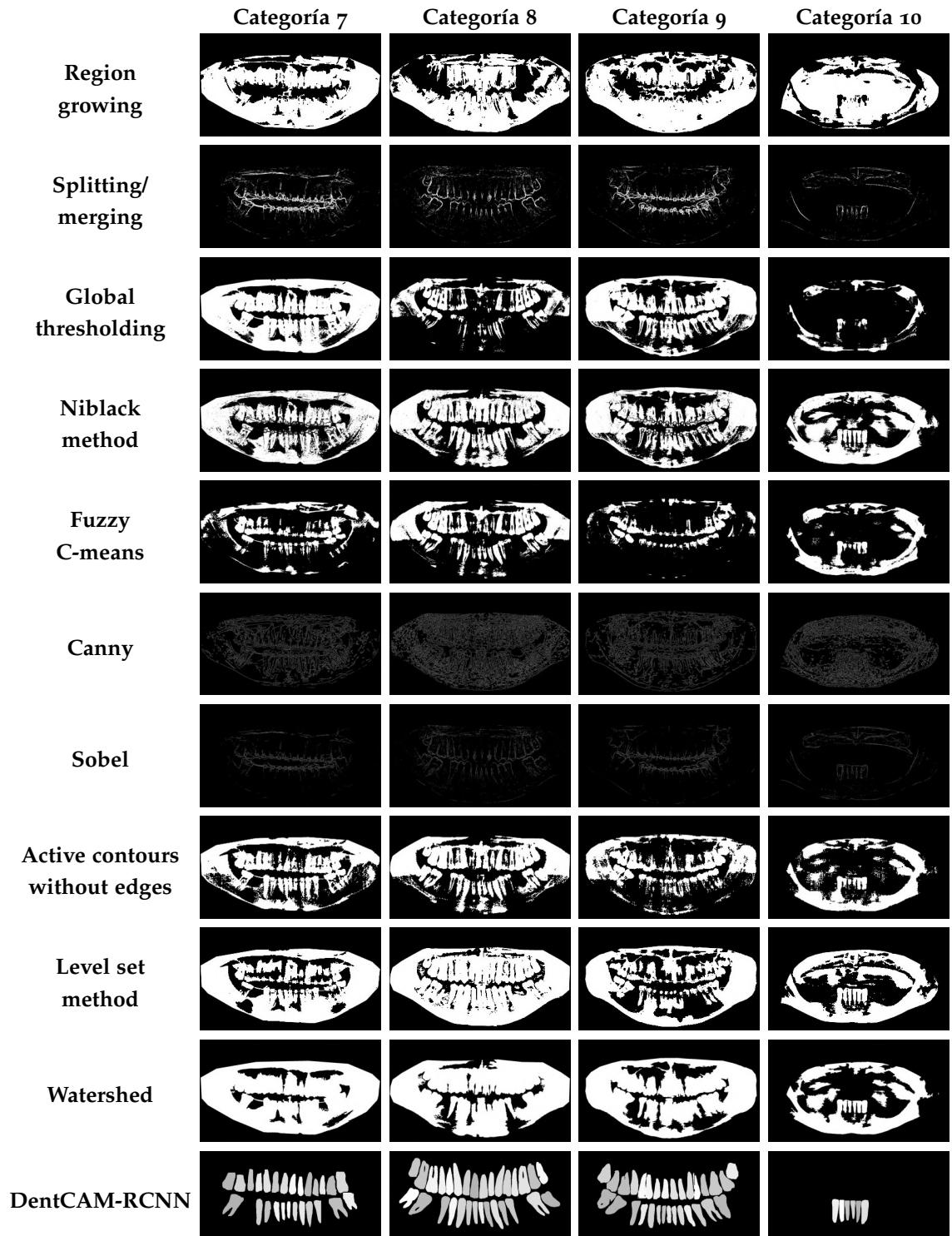


Figura 10.12.: Comparación cualitativa entre métodos clásicos de segmentación no supervisados y DentCAM-RCNN sobre las categorías 7 a 10.

Estudio	Imágenes	F1-Score	DSC	IoU
Jader et al. [JFR ⁺ 18]	193	0.8800	-	-
Lee et al. [LHK ⁺ 20]	1024	0.8750	-	0.8770
Brahmi y Jdey [BJ23]	107	0.6300	-	-
Rubiu et al. [RBC ⁺ 23]	1000	-	0.8700	-
DentCAM-RCNN	543	0.9604	0.8776	0.7923

Tabla 10.6.: Comparativa de métricas de segmentación dental (F1-Score, DSC e IoU) entre distintos estudios recientes y DentCAM-RCNN, haciendo uso ambos de la arquitectura Mask R-CNN. Se incluyen el número de imágenes empleadas en cada estudio y se destacan en negrita los mejores resultados obtenidos para cada métrica.

11. Conclusiones y Trabajos Futuros

Este Trabajo de Fin de Grado ha abordado el problema de la segmentación de instancias dentales en radiografías panorámicas mediante la arquitectura Mask R-CNN, integrando técnicas de explicabilidad *post-hoc* para mejorar la explicabilidad del modelo. La propuesta desarrollada, denominada DentCAM-RCNN, consiste en el ajuste fino de Mask R-CNN al dominio dental (usando el conjunto de datos DNS Panoramic Images) junto con la implementación de Grad-CAM como herramienta de análisis visual. Los objetivos principales han sido adaptar Mask R-CNN al dominio dental mediante ajuste fino, evaluar cuantitativamente su rendimiento en tareas de segmentación, e implementar Grad-CAM como herramienta de interpretación visual para analizar las decisiones del modelo.

DentCAM-RCNN ha logrado resultados significativamente superiores al modelo base pre-entrenado en COCO. En concreto, se ha obtenido un F1-Score del 96.04 %, una precisión del 97.36 %, un DSC del 87.76 % y un IoU del 79.23 % en promedio en la partición de test. Estos resultados superan ampliamente tanto a métodos clásicos no supervisados como a diversos enfoques recientes basados en aprendizaje profundo para la segmentación dental, lo que sitúa al modelo desarrollado como una opción sólida y eficaz para tareas clínicas de segmentación en imágenes de radiografías dentales panorámicas.

Desde el punto de vista de la explicabilidad, se ha implementado con éxito una adaptación de Grad-CAM para la arquitectura Mask R-CNN, basada en trabajos previos de Inbaraj et al. [XJ₂₁, XVM⁺₂₁, XVM⁺₂₂]. La evaluación cuantitativa, basada en métricas como Deletion, Insertion, Stability e IoU_{Grad-CAM}, ha mostrado diferencias notables entre los modelos, reflejando que DentCAM-RCNN centra su atención en regiones relevantes, mientras que el modelo base lo hace en zonas menos informativas. Esto no implica una mayor o menor eficacia de Grad-CAM, sino que confirma su utilidad como herramienta diagnóstica al evidenciar cuándo un modelo toma decisiones basadas en características poco adecuadas. Esta integración ha permitido incorporar mecanismos de explicabilidad a Mask R-CNN, una arquitectura originalmente diseñada sin mecanismos interpretables, facilitando así su análisis, validación y posible uso en el análisis automático de radiografías dentales panorámicas.

Asimismo, se ha realizado una comparación cualitativa con múltiples métodos clásicos, mostrando que DentCAM-RCNN produce máscaras más coherentes, precisas y generalizables. En conjunto, estos resultados validan empíricamente la eficacia del ajuste fino y la utilidad de las técnicas de explicabilidad para aumentar la confianza en los sistemas de IA aplicados al ámbito médico.

Los objetivos específicos establecidos al inicio del trabajo se han cumplido en su totalidad:

1. Se ha analizado y comprendido el funcionamiento interno de Mask R-CNN en tareas de segmentación de instancias.
2. Se ha investigado el campo de la XAI y la técnica Grad-CAM.

11. Conclusiones y Trabajos Futuros

3. Se ha implementado Grad-CAM sobre Mask R-CNN en el contexto de imágenes dentales.
4. Se ha evaluado la utilidad de esta integración mediante métricas cuantitativas y análisis cualitativos.

El código y los experimentos han sido compartidos públicamente en dos repositorios de GitHub para garantizar la transparencia y reproducibilidad: <https://github.com/juanmaarg6/TFG> y https://github.com/juanmaarg6/DentCAM_RCNN.

A partir de los resultados obtenidos, se identifican las siguientes líneas de trabajo futuro:

- Evaluar otras técnicas de explicabilidad (como LIME [RSG16] o SHAP [LL17]) y compararlas con Grad-CAM en tareas de segmentación.
- Adaptar la metodología a otras arquitecturas de segmentación de instancias como HTC [CPW⁺19], PANet [LQQ⁺18] o ResNeSt [ZWZ⁺20].
- Realizar estudios con expertos clínicos para validar la utilidad práctica de los mapas de atención generados.
- Automatizar la generación e interpretación de mapas explicativos para facilitar su uso en entornos reales.

A. Apéndice: Evaluación detallada del modelo base

En este apéndice se presentan los resultados detallados por categoría dental obtenidos durante la evaluación del modelo base de Mask R-CNN preentrenado con los pesos del conjunto COCO, sin ningún ajuste al dominio dental. La [Tabla A.1](#) recoge métricas de detección (Precision, Recall, F1-Score, AP@50) y segmentación (DSC, IoU, HD) para las particiones de entrenamiento, validación y test, mientras que la [Tabla A.2](#) presenta las métricas de explicabilidad (Deletion, Insertion, Stability e IoU_{Grad-CAM}) obtenidas al aplicar Grad-CAM sobre las predicciones del modelo. Estos resultados permiten analizar las limitaciones del modelo base al abordar imágenes dentales sin ajuste específico al dominio.

A. Apéndice: Evaluación detallada del modelo base

Categoría	Partición	Precision	Recall	F1-Score	AP@50	DSC	IoU	HD
1	Train	0.0429	0.0467	0.0447	0.0049	0.0179	0.0140	28.5204
	Validation	0.0286	0.0310	0.0297	0.0048	0.0133	0.0102	22.8830
	Test	0.0190	0.0206	0.0198	0.0008	0.0217	0.0169	23.6414
2	Train	0.0602	0.0624	0.0613	0.0112	0.0144	0.0114	26.0040
	Validation	0.0741	0.0763	0.0751	0.0161	0.0178	0.0141	25.4117
	Test	0.0659	0.0700	0.0679	0.0134	0.0173	0.0136	28.5278
3	Train	0.0660	0.0682	0.0671	0.0132	0.0212	0.0170	24.0653
	Validation	0.0735	0.0766	0.0750	0.0118	0.0139	0.0110	29.3295
	Test	0.0429	0.0452	0.0440	0.0070	0.0105	0.0081	29.4597
4	Train	0.0637	0.0660	0.0649	0.0154	0.0199	0.0158	25.7580
	Validation	0.0633	0.0686	0.0658	0.0169	0.0159	0.0126	27.1136
	Test	0.0551	0.0588	0.0569	0.0099	0.0119	0.0095	23.4994
7	Train	0.0524	0.0659	0.0584	0.0096	0.0084	0.0067	25.9072
	Validation	0.0381	0.0503	0.0434	0.0047	0.0041	0.0033	35.5582
	Test	0.0629	0.0846	0.0721	0.0111	0.0045	0.0034	41.6355
8	Train	0.0584	0.0765	0.0662	0.0124	0.0107	0.0084	31.9171
	Validation	0.0471	0.0629	0.0539	0.0099	0.0084	0.0066	27.1450
	Test	0.0526	0.0662	0.0586	0.0130	0.0099	0.0078	30.7589
9	Train	0.0571	0.0674	0.0619	0.0083	0.0100	0.0078	27.2674
	Validation	0.0190	0.0290	0.0230	0.0314	0.0180	0.0147	28.6720
	Test	0.0857	0.0984	0.0916	0.0124	0.0166	0.0129	31.1952
10	Train	0.0658	0.0822	0.0731	0.0142	0.0121	0.0095	28.1247
	Validation	0.0476	0.0877	0.0617	0.0139	0.0088	0.0068	27.8894
	Test	0.0343	0.0526	0.0415	0.0184	0.0073	0.0056	39.0669
Media	Train	0.0583	0.0669	0.0626	0.0111	0.0143	0.0115	27.5705
	Validation	0.0490	0.0603	0.0542	0.0137	0.0125	0.0099	27.8877
	Test	0.0523	0.0628	0.0571	0.0109	0.0125	0.0097	30.7306

Tabla A.1.: Métricas de detección y segmentación por categoría y partición del modelo base de Mask R-CNN preentrenado con los pesos de COCO. La tabla incluye métricas de detección (Precision, Recall, F1-Score, AP@50) y segmentación (DSC, IoU, HD) para cada categoría dental analizada. Se desglosan los resultados por conjuntos de entrenamiento, validación y test, e incluye una fila final con la media aritmética de todas las categorías. Los valores reflejan un rendimiento muy limitado sin ajuste al dominio, con métricas de segmentación especialmente bajas y errores significativos en la localización y forma de las máscaras.

Categoría	Partición	Deletion	Insertion	Stability	$\text{IoU}_{\text{Grad-CAM}}$
1	Train	0.9964	0.9949	0.3193	0.1783
	Validation	0.9969	0.9947	0.2273	0.1352
	Test	0.9972	0.9949	0.2383	0.1428
2	Train	0.9938	0.9943	0.2610	0.1526
	Validation	0.9944	0.9939	0.2354	0.1402
	Test	0.9949	0.9936	0.2491	0.1459
3	Train	0.9831	0.9831	0.2773	0.1261
	Validation	0.9827	0.9837	0.1548	0.1542
	Test	0.9851	0.9850	0.3689	0.1108
4	Train	0.9954	0.9957	0.2124	0.1380
	Validation	0.9956	0.9951	0.2649	0.1067
	Test	0.9954	0.9964	0.2719	0.1328
7	Train	0.9920	0.9931	0.1886	0.1187
	Validation	0.9930	0.9929	0.1725	0.1058
	Test	0.9935	0.9932	0.1959	0.1140
8	Train	0.9918	0.9924	0.2121	0.1271
	Validation	0.9925	0.9927	0.2013	0.1199
	Test	0.9929	0.9925	0.2208	0.1226
9	Train	0.9941	0.9937	0.2740	0.1602
	Validation	0.9952	0.9933	0.2457	0.1373
	Test	0.9956	0.9936	0.2669	0.1496
10	Train	0.9947	0.9940	0.2835	0.1648
	Validation	0.9951	0.9938	0.2616	0.1529
	Test	0.9955	0.9939	0.2724	0.1591
Media	Train	0.9929	0.9932	0.2564	0.1457
	Validation	0.9932	0.9933	0.2218	0.1316
	Test	0.9938	0.9936	0.2472	0.1347

Tabla A.2.: Métricas de explicabilidad por categoría y partición del modelo base de Mask R-CNN preentrenado con los pesos de COCO. La tabla recoge las métricas Deletion, Insertion, Stability e $\text{IoU}_{\text{Grad-CAM}}$ obtenidas al aplicar Grad-CAM sobre las predicciones del modelo en los conjuntos de entrenamiento, validación y test. Los resultados evidencian una atención poco consistente y mal localizada del modelo base sobre las estructuras dentales.

B. Apéndice: Evaluación detallada del modelo ajustado

En este apéndice se presentan los resultados detallados por categoría dental obtenidos durante la evaluación del modelo de Mask R-CNN ajustado al dominio dental. La [Tabla B.1](#) recoge las métricas de detección (Precision, Recall, F1-Score, AP@50) y segmentación (DSC, IoU, HD) para los conjuntos de entrenamiento, validación y test, mientras que la [Tabla B.2](#) muestra las métricas de explicabilidad (Deletion, Insertion, Stability e IoU_{Grad-CAM}) calculadas a partir de la aplicación de Grad-CAM sobre las predicciones del modelo. Estos resultados permiten analizar en profundidad el comportamiento del modelo tras su adaptación al contexto específico de imágenes dentales, evidenciando una mejora tanto en rendimiento como en interpretabilidad. La evolución de las funciones de pérdida durante el entrenamiento puede observarse en la [Figura B.1](#).

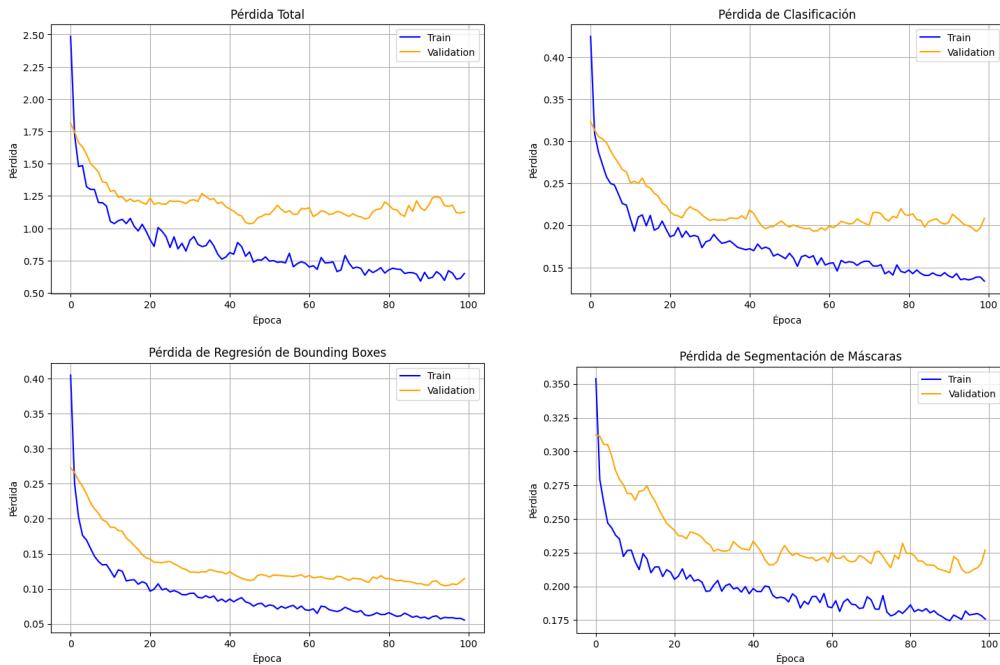


Figura B.1.: Evolución de las funciones de pérdida durante el entrenamiento del modelo de Mask R-CNN con ajuste en las capas superiores para adaptar los pesos preentrenados de COCO al dominio dental. Se representan, de arriba abajo: la pérdida total $\mathcal{L}_{\text{total}}$, la pérdida de clasificación \mathcal{L}_{cls} , la pérdida de regresión de *bounding boxes* $\mathcal{L}_{\text{bbox}}$ y la pérdida de segmentación de máscaras $\mathcal{L}_{\text{mask}}$.

B. Apéndice: Evaluación detallada del modelo ajustado

Categoría	Partición	Precision	Recall	F1-Score	AP@50	DSC	IoU	HD
1	Train	0.9903	0.9883	0.9893	0.9883	0.8997	0.8198	9.7224
	Validation	0.9323	0.9612	0.9466	0.9599	0.9074	0.8322	9.3436
	Test	0.9394	0.9588	0.9490	0.9511	0.9051	0.8288	10.0224
2	Train	0.9912	0.9519	0.9711	0.9553	0.9190	0.8520	8.9095
	Validation	0.9663	0.9052	0.9348	0.9016	0.9040	0.8272	11.0629
	Test	0.9880	0.9568	0.9721	0.9553	0.9087	0.8345	10.8911
3	Train	0.9863	0.9501	0.9678	0.9534	0.8978	0.8173	10.9391
	Validation	0.9825	0.9574	0.9698	0.9633	0.8973	0.8165	11.6764
	Test	0.9895	0.9447	0.9666	0.9454	0.8976	0.8172	11.7590
4	Train	0.9937	0.9399	0.9661	0.9494	0.9169	0.8483	9.5926
	Validation	0.9357	0.9336	0.9347	0.9252	0.8686	0.7706	16.0321
	Test	0.9776	0.9521	0.9647	0.9497	0.8640	0.7628	17.5789
7	Train	0.9954	0.9746	0.9849	0.9776	0.8657	0.7661	10.9918
	Validation	0.9872	0.9686	0.9778	0.9692	0.8546	0.7500	13.4831
	Test	0.9609	0.9462	0.9535	0.9399	0.8362	0.7218	14.6092
8	Train	0.9742	0.9622	0.9682	0.9627	0.9061	0.8299	10.9782
	Validation	0.9579	0.9543	0.9561	0.9157	0.8830	0.7930	13.6069
	Test	0.9663	0.9206	0.9429	0.9244	0.8773	0.7844	14.9395
9	Train	0.9961	0.9588	0.9771	0.9609	0.8866	0.7987	11.0924
	Validation	1.0000	1.0000	1.0000	1.0000	0.8741	0.7792	12.4019
	Test	0.9508	0.9508	0.9508	0.9471	0.8865	0.8007	12.5951
10	Train	0.9921	0.9783	0.9852	0.9802	0.9212	0.8555	8.8819
	Validation	0.9826	0.9912	0.9869	0.9889	0.9185	0.8507	8.2976
	Test	1.0000	0.9737	0.9867	0.9790	0.9176	0.8489	9.7866
Media	Train	0.9869	0.9620	0.9741	0.9585	0.9017	0.8123	10.1341
	Validation	0.9606	0.9528	0.9566	0.9479	0.8847	0.7952	11.5458
	Test	0.9736	0.9480	0.9604	0.9398	0.8776	0.7923	12.2605

Tabla B.1.: Métricas de detección y segmentación por categoría y partición del modelo de Mask R-CNN con ajuste en las capas superiores para adaptar los pesos preentrenados de COCO al dominio dental. La tabla recoge métricas tanto de detección (Precision, Recall, F1-Score, AP@50) como de segmentación (DSC, IoU, HD) para cada categoría dental en los conjuntos de entrenamiento, validación y test. Se incluye una fila final con la media aritmética de todas las categorías. Los resultados reflejan una mejora sustancial respecto al modelo base, con valores superiores al 90 % en la mayoría de métricas y categorías. Las métricas de segmentación muestran una alta coincidencia entre las máscaras predichas y las reales, lo que evidencia una adaptación exitosa al dominio dental tras el ajuste fino.

Categoría	Partición	Deletion	Insertion	Stability	$\text{IoU}_{\text{GradCAM}}$
1	Train	0.1567	0.9966	0.8822	0.4258
	Validation	0.1528	0.9990	0.8999	0.4277
	Test	0.1455	0.9899	0.8950	0.4518
2	Train	0.1594	0.9974	0.8827	0.4201
	Validation	0.1582	0.9969	0.8869	0.4182
	Test	0.1518	0.9963	0.8901	0.4246
3	Train	0.1876	0.9980	0.8779	0.4242
	Validation	0.1934	0.9981	0.8762	0.4105
	Test	0.1703	0.9981	0.8827	0.4290
4	Train	0.1782	0.9965	0.8732	0.4193
	Validation	0.1790	0.9963	0.8727	0.4277
	Test	0.1865	0.9971	0.8703	0.4262
7	Train	0.1358	0.9946	0.8896	0.4116
	Validation	0.1455	0.9978	0.8841	0.3949
	Test	0.1544	0.9965	0.8923	0.4051
8	Train	0.1542	0.9935	0.8908	0.4416
	Validation	0.1260	0.9846	0.8990	0.4390
	Test	0.1639	0.9939	0.9039	0.4863
9	Train	0.1642	0.9961	0.8810	0.4289
	Validation	0.1478	0.9954	0.8845	0.4223
	Test	0.1563	0.9968	0.8862	0.4291
10	Train	0.1539	0.9982	0.8877	0.4344
	Validation	0.1384	0.9960	0.8892	0.4310
	Test	0.1471	0.9976	0.8903	0.4379
Media	Train	0.1612	0.9964	0.8806	0.4258
	Validation	0.1577	0.9963	0.8866	0.4214
	Test	0.1595	0.9965	0.8891	0.4365

Tabla B.2.: Métricas de explicabilidad por categoría y partición del modelo de Mask R-CNN con ajuste en las capas superiores para adaptar los pesos preentrenados de COCO al dominio dental (DentCAM-RCNN). La tabla recoge las métricas Deletion, Insertion, Stability e $\text{IoU}_{\text{Grad-CAM}}$ obtenidas al aplicar Grad-CAM sobre las predicciones del modelo en los conjuntos de entrenamiento, validación y test. Los resultados muestran una atención mucho más coherente, estable y centrada en las regiones relevantes, reflejando una mayor calidad explicativa del modelo tras su adaptación al dominio dental.

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.
- [ADRS⁺19] A. Barredo Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins, R. Chatila, y F. Herrera. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. 2019.
- [AEZ15] R. B. Ali, R. Ejbali, y M. Zaied. Gpu-based segmentation of dental x-ray images using active contours without edges. *International Conference on Intelligent Systems Design and Applications (ISDA)*, 1:505–510, 2015.
- [Als15] M. K. Alsmadi. A hybrid fuzzy c-means and neutrosophic for jaw lesions segmentation. *Ain Shams Engineering Journal*, 2015.
- [AM14] J. Andén y S. Mallat. Deep scattering spectrum. *IEEE Transactions on Signal Processing*, 62(16):4114–4128, 2014.
- [AMMIL12] Y. S. Abu-Mostafa, M. Magdon-Ismail, y H. Lin. *Learning From Data*. AMLBook, 2012.
- [AMVAo8] J. P. Antoine, R. Murenzi, P. Vandergheynst, y S. T. Ali. Two-dimensional wavelets and their relatives. *Cambridge University Press*, 2008.
- [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.
- [BB23] C. M. Bishop y H. Bishop. *Deep Learning: Foundations and Concepts*. Springer Nature, 2023.
- [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.
- [Biso06] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.
- [BJ23] W. Brahmi y I. Jdey. Automatic tooth instance segmentation and identification from panoramic x-ray images using deep cnn. *Multimedia Tools and Applications*, 83(18):55565–55585, 2023.
- [BM13] J. Bruna y S. Mallat. Invariant scattering convolution networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1872–1886, 2013.
- [BPL10] Y.-L. Boureau, J. Ponce, y Y. LeCun. A theoretical analysis of feature pooling in visual recognition. *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, páginas 111–118, 2010.

Bibliografía

- [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.
- [CPW⁺19] K. Chen, J. Pang, J. Wang, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Shi, W. Ouyang, C. Change Loy, y D. Lin. Hybrid task cascade for instance segmentation. *arXiv preprint arXiv:1901.07518*, 2019.
- [DHL⁺16] J. Dai, K. He, Y. Li, S. Ren, y J. Sun. Instance-sensitive fully convolutional networks. *CoRR*, abs/1603.08678, 2016.
- [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. Kadoury, y C. Pal. The importance of skip connections in biomedical image segmentation. *CoRR*, abs/1608.04117, 2016.
- [DVK17a] F. Doshi-Velez y B. Kim. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608*, 2017.
- [DVK17b] F. Doshi-Velez y B. Kim. Towards a rigorous science of interpretable machine learning. *arXiv*, 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.
- [Fre14] A. A. Freitas. Comprehensible classification models: a position paper. *ACM SIGKDD Explorations Newsletter*, 15(1):1–10, 2014.
- [FV17] R. Fong y A. Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. páginas 3429–3437, 2017.
- [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.
- [Gla25] Glassdoor. Sueldos para el puesto de research scientist en españa. https://www.glassdoor.es/Sueldos/research-scientist-sueldo-SRCH_K00,19.htm, 2025.
- [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.
- [HHo8] C. H. Huang y C. Y. Hsu. Computer-assisted orientation of dental periapical radiographs to the occlusal plane. *Oral Surgery, Oral Medicine, Oral Pathology, Oral Radiology and Endodontology*, 105(5):649–653, 2008.
- [HLo6] 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.
- [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.
- [JFR⁺18] G. Jader, J. Fontineli, M. Ruiz, K. Abdalla, M. Pithon, y L. Oliveira. Deep instance segmentation of teeth in panoramic x-ray images. *Conference on Graphics, Patterns and Images (SIBGRAPI)*, páginas 400–407, 2018.
- [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.
- [JS08] V. Jain y S. Seung. Natural image denoising with convolutional networks. *Advances in Neural Information Processing Systems*, 21, 2008.
- [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.

Bibliografía

- [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.
- [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.
- [LBC⁺24] L. Longo, M. Brcic, F. Cabitza, J. Choi, R. Confalonieri, J. Del Ser, R. Guidotti, Y. Hayashi, F. Herrera, A. Holzinger, R. Jiang, H. Khosravi, F. Lecue, G. Malgieri, A. Páez, W. Samek, J. Schneider, T. Speith, y S. Stumpf. Explainable artificial intelligence (xai) 2.0: A manifesto of open challenges and interdisciplinary research directions. *Information Fusion*, 106:102301, 2024.
- [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.
- [LHK⁺20] J.-H. Lee, S.-S. Han, Y.-H. Kim, C. Lee, y I. Kim. Application of a fully deep convolutional neural network to the automation of tooth segmentation on panoramic radiographs. *Oral Surg. Oral Med. Oral Pathol. Oral Radiol.*, 129(6):635–642, 2020.
- [Liu18] Y. Liu. The confusing metrics of ap and map for object detection / instance segmentation. <https://yanfengliu.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.
- [LQQ⁺18] S. Liu, L. Qi, H. Qin, J. Shi, y J. Jia. Path aggregation network for instance segmentation. *arXiv preprint arXiv:1803.01534*, 2018.
- [LSD14] J. Long, E. Shelhamer, y T. Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014.
- [LSSL12] H. Li, G. Sun, H. Sun, y W. Liu. Watershed algorithm based on morphology for dental x-ray images segmentation. *International Conference on Signal Processing Proceedings*, 2:877–880, 2012.
- [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.
- [MD11] C. K. Modi y N. P. Desai. A simple and novel algorithm for automatic selection of roi for dental radiograph segmentation. *Canadian Conference on Electrical and Computer Engineering (CCECE)*, páginas 000504–000507, 2011.
- [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.
- [Mol20] C. Molnar. *Interpretable Machine Learning*. Lulu.com, 2020.
- [MRW19] B. Mittelstadt, C. Russell, y S. Wachter. Explaining explanations in ai. *Communications of the ACM*, 61(5):56–65, 2019.
- [MSM18] G. Montavon, W. Samek, y K. R. Müller. Methods for interpreting and understanding deep neural networks. *Digital Signal Processing*, 73:1–15, 2018.
- [Mur22] K. P. Murphy. *Probabilistic Machine Learning: An Introduction*. MIT Press, 2022.
- [Mur23] K. P. Murphy. *Probabilistic Machine Learning: Advanced Topics*. MIT Press, 2023.
- [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.
- [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.

Bibliografía

- [Oua17] A. Ouaknine. Review of deep learning algorithms for image classification. <https://medium.com/zylapp/review-of-deep-learning-algorithms-for-image-classification-5fd8ca4a05e2>, 2017.
- [PCDo8] 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.
- [PDS18] V. Petruik, A. Das, y K. Saenko. Rise: Randomized input sampling for explanation of black-box models. *arXiv*, 2018.
- [Pin17] J. Pinto. Masknet: An instance segmentation algorithm. *Chalmers University*, 2017.
- [Pre05] R. S. Pressman. *Software Engineering: A Practitioner’s Approach*. Palgrave Macmillan, 2005.
- [Pri23] S. J. D. Prince. *Understanding Deep Learning*. MIT Press, 2023.
- [PY10] Sinno Jialin Pan y Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.
- [RAH⁺15] M. R. M. Razali, N. S. Ahmad, R. Hassan, Z. M. Zaki, y W. Ismail. Sobel and canny edges segmentations for the dental age assessment. *International Conference on Computer Assisted System in Health (CASH)*, páginas 62–66, 2015.
- [RAZI14] M. R. Mohamed Razali, N. S. Ahmad, Z. Mohd Zaki, y W. Ismail. Region of adaptive threshold segmentation between mean, median and otsu threshold for dental age assessment. *International Conference on Computer, Communications, and Control Technology (I4CT)*, páginas 353–356, 2014.
- [RBC⁺23] G. Rubiu, M. Bologna, M. Cellina, M. Cè, D. Sala, R. Pagani, E. Mattavelli, D. Fazzini, S. Ibba, S. Papa, y M. Alì. Teeth segmentation in panoramic dental x-ray using mask regional convolutional neural network. *Appl. Sci.*, 13(13):7947, 2023.
- [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.
- [RN22] S. Russell y P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, 4 edición, 2022.
- [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.
- [RRN13] A. Ehsani Rad, M. S. Mohd Rahim, y A. Norouzi. Digital dental x-ray image segmentation and feature extraction. *TELKOMNIKA Indonesian Journal of Electrical Engineering*, 11(6):3109–3114, 2013.

- [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.
- [Rud16] S. Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv: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/11_9-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.
- [SOP18] G. Silva, L. Oliveira, y M. Pithon. Automatic segmenting teeth in x-ray images: Trends, a novel data set, benchmarking and future perspectives. *Expert Systems with Applications*, 107:15–31, 2018.
- [Sor18] M. Sorsater. Active learning for road segmentation using convolutional neural networks. *Linkoping University*, 2018.
- [SPA14] R. B. Subramanyam, K. P. Prasad, y B. Anuradha. Different image segmentation techniques for dental image extraction. *International Journal of Engineering Research and Applications (IJERA)*, 4(7):173–177, 2014.
- [SPOP20] B. Silva, L. Pinheiro, L. Oliveira, y M. Pithon. A study on tooth segmentation and numbering using end-to-end deep neural networks. *Conference on Graphics, Patterns and Images (SIBGRAPI)*, 2020.
- [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.
- [Sze22] R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer Nature, 2022.
- [TIF24] A. Torralba, P. Isola, y W. T. Freeman. *Foundations of Computer Vision*. MIT Press, 2024.
- [Uni20] Stanford University. Cs231n: Convolutional neural networks for visual recognition. <https://cs231n.stanford.edu/2020/syllabus.html>, 2020.

Bibliografía

- [Vai93] P. P. Vaidyanathan. *Multirate Systems and Filter Banks*. Prentice Hall, 1993.
- [Vin24] A. Vina. ¿qué es r-cnn? un resumen rápido. <https://www.ultralytics.com/es/blog/what-is-r-cnn-a-quick-overview>, 2024.
- [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.
- [ZBPG22] A. Zhukov, J. Benois-Pineau, y R. Giot. Evaluation of explanation methods of ai – cnns in image classification tasks with reference-based and no-reference metrics. *arXiv*, 2022.
- [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.
- [ZWZ⁺20] H. Zhang, C. Wu, Z. Zhang, Y. Zhu, H. Lin, Z. Zhang, Y. Sun, T. He, J. Mueller, R. Manmatha, M. Li, y A. Smola. Resnest: Split-attention networks. *arXiv preprint arXiv:2004.08955*, 2020.
- [ZZXW18] Z. Zhao, P. Zheng, S. Xu, y X. Wu. Object detection with deep learning: A review. *CoRR*, abs/1807.05511, 2018.