

UNIVERSIDAD MIGUEL HERNÁNDEZ DE ELCHE
ESCUELA POLITÉCNICA SUPERIOR DE ELCHE
GRADO EN INGENIERÍA ELECTRÓNICA Y
AUTOMÁTICA INDUSTRIAL



Biblioteca

“ANÁLISIS COMPARATIVO DE
DESCRIPTORES CLÁSICOS Y
EMBEDDINGS DE RESNET-50:
EVALUACIÓN EN TEXTURAS
COMPLEJAS”

TRABAJO FIN DE GRADO

Diciembre - 2025

AUTOR: Ángel Andrés Medrano Villasmil

DIRECTOR: Luis Miguel Jiménez García

RESUMEN

En el presente Trabajo de Fin de Grado se aborda el problema de la clasificación de texturas complejas mediante visión artificial. Se realiza una comparación exhaustiva entre métodos tradicionales de extracción de características (LBP, GLCM, Gabor) y técnicas modernas basadas en aprendizaje profundo, específicamente utilizando los embeddings generados por la arquitectura ResNet-50.

Los experimentos se llevan a cabo sobre el dataset DTD (Describable Textures Dataset), evaluando no solo el rendimiento en términos de precisión (accuracy), sino también la eficiencia computacional, así como la explicabilidad de los modelos de Deep Learning. Los resultados demuestran las fortalezas y debilidades de cada enfoque, proporcionando una guía para la selección de algoritmos en aplicaciones industriales de inspección visual.



ÍNDICE

RESUMEN EJECUTIVO	I
--------------------------	----------

ÍNDICE DE TABLAS	IX
-------------------------	-----------

ÍNDICE DE FIGURAS	XIII
--------------------------	-------------

1. Introducción	1
------------------------	----------

1.1. Contexto y motivación	1
--------------------------------------	---

1.2. Problema de investigación	2
--	---

1.3. Preguntas de investigación y enfoque del estudio	3
---	---

1.4. Objetivos	5
--------------------------	---

1.4.1. Objetivo general	5
-----------------------------------	---

1.4.2. Objetivos específicos	5
--	---

1.5. Alcance y limitaciones	6
---------------------------------------	---

1.6. Estructura de la memoria	8
---	---

2. Fundamentos teóricos y estado del arte	10
--	-----------

2.1. Visión por computador y clasificación de texturas	10
--	----

2.1.1. Concepto de textura y propiedades relevantes	12
---	----

2.1.2. Aplicaciones de la clasificación de texturas en automatización	13
---	----

2.2. Descriptores clásicos de textura	15
---	----

2.2.1. Local Binary Patterns (LBP) y variantes	16
--	----

2.2.2. Características Haralick y matrices GLCM	18
---	----

2.2.3. Filtros de Gabor y bancos multi-escala/multi-orientación . . .	20
---	----

2.3.	Clasificadores tradicionales para descriptores de textura	22
2.3.1.	Máquinas de Vectores de Soporte (SVM)	23
2.3.2.	Otros clasificadores (k-NN, Random Forest, etc.) como contexto	26
2.4.	Redes neuronales convolucionales (CNN): fundamentos	29
2.4.1.	Arquitectura básica de una CNN	30
2.4.2.	Entrenamiento y aprendizaje de representaciones jerárquicas .	32
2.4.3.	Ventajas y limitaciones en el contexto de texturas y automa- tización	33
2.5.	Arquitecturas convolucionales de última generación (SOTA)	35
2.5.1.	Evolución de las arquitecturas CNN SOTA	35
2.5.2.	Redes residuales (ResNet)	36
2.6.	Aprendizaje transferido y fine-tuning en visión por computador . . .	38
2.6.1.	Concepto general de aprendizaje transferido en CNN	39
2.6.2.	Estrategias habituales: extracción de características vs fine- tuning	40
2.7.	CNN como extractores de características: embeddings y capas inter- medias	42
2.7.1.	Embeddings derivados de capas intermedias y finales	42
2.8.	Explicabilidad en visión por computador	43
2.8.1.	Grad-CAM y variantes	43
2.8.2.	Otras técnicas de XAI visual	44
2.9.	Dataset de trabajo: Describable Textures Dataset (DTD)	45
3.	Metodología y diseño experimental	47
3.1.	Justificación metodológica del enfoque basado en un clasificador uni- ficado (SVM)	47

3.2. Bloques de descriptores y esquema comparativo	48
3.2.1. Bloque 1 – Descriptores clásicos básicos (LBP, Haralick, Gabor)	48
3.2.2. Bloque 2 – Embeddings de ResNet-50 (feature extraction congelado)	48
3.2.3. Bloque 3 – Embeddings de ResNet-50 con fine-tuning	49
3.3. Preparación de datos y particionado del DTD	49
3.4. Escenarios experimentales	49
3.5. Criterios de evaluación y métricas	50
3.5.1. Métrica de rendimiento principal: <i>accuracy</i>	50
3.5.2. Análisis de errores: matrices de confusión	51
3.5.3. Métricas de eficiencia computacional	52
3.5.4. Criterios de explicabilidad	52
3.6. Protocolo experimental común	53
3.7. Reproducibilidad y trazabilidad de los experimentos	54
4. Desarrollo experimental	56
4.1. Entorno hardware y software	56
4.2. Implementación del pipeline de descriptores clásicos	56
4.3. Implementación del pipeline basado en CNN	57
4.3.1. Entrenamiento de ResNet-50	57
4.3.2. Extracción de embeddings de ResNet-50	57
4.4. Extracción de descriptores y entrenamiento de la SVM	57
4.5. Organización práctica de los experimentos	58
5. Resultados	59

5.1. Rendimiento global y comparación de métodos	59
5.2. Análisis de convergencia y entrenamiento	60
5.3. Análisis de confusiones y errores	61
5.4. Resultados de explicabilidad (Grad-CAM)	62
6. Discusión	64
6.1. La brecha entre descriptores clásicos y profundos	64
6.2. Robustez y estabilidad del entrenamiento	64
6.3. Eficiencia vs. precisión: implicaciones prácticas	65
7. Conclusiones y respuestas a las preguntas de investigación	66
7.1. ¿Superan las CNN a los descriptores clásicos en rendimiento?	66
7.2. ¿Cómo se comportan frente a los errores?	66
7.3. ¿Es viable computacionalmente en automatización?	67
7.4. ¿Qué aporta la explicabilidad?	67
8. Trabajo futuro	69
A. Código fuente y desarrollo software	70
A.1. Orquestación y reproducibilidad	70
A.2. Implementación de descriptores clásicos	71
A.3. Arquitectura y entrenamiento de modelos <i>Deep Learning</i>	72
A.4. Pipeline de clasificación unificado (SVM)	74
B. Métricas detalladas de los experimentos	75
B.1. Descriptores clásicos + SVM	76
B.2. Embeddings de ResNet-50 + SVM	77

B.3. ResNet-50 End-to-End (Softmax)	77
---	----

BIBLIOGRAFÍA	79
---------------------	-----------



ÍNDICE DE TABLAS

5.1. Resumen comparativo de rendimiento (promedio de 10 <i>splits</i>). El <i>Gap</i> representa la diferencia entre <i>accuracy</i> en entrenamiento y test, actuando como indicador de sobreajuste.	59
B.1. Métricas detalladas por <i>split</i> para GLCM + SVM.	76
B.2. Métricas detalladas por <i>split</i> para Gabor + SVM.	76
B.3. Métricas detalladas por <i>split</i> para LBP multiescala + SVM.	76
B.4. Métricas detalladas por <i>split</i> para ResNet50 (Frozen) + SVM.	77
B.5. Métricas detalladas por <i>split</i> para ResNet50 (Finetune) + SVM.	77
B.6. Métricas detalladas por <i>split</i> para ResNet50 (Frozen) End-to-End.	78
B.7. Métricas detalladas por <i>split</i> para ResNet50 (Finetune) End-to-End.	78



ÍNDICE DE FIGURAS

2.1. Flujo habitual de un sistema de visión por computador	10
2.2. Comparativa visual: Textura natural recortada (izq) frente a patrón artificial (der).	11
2.3. Visualización de la transformación LBP. A la izquierda, la textura original con patrones circulares oscuros. A la derecha, el mapa de códigos LBP resultante ($P = 8, R = 1$, uniforme), donde se aprecia cómo el descriptor resalta los bordes y la estructura local de la textura ignorando la información de color global.	17
2.4. Representación de vecindarios circulares LBP extendidos para diferentes resoluciones (P, R). De izquierda a derecha, se muestran ejemplos variando el número de puntos de muestreo (P) y el radio (R), utilizando interpolación bilineal para los puntos que no coinciden con la rejilla.	18
2.5. Visualización de una Matriz de Coocurrencia (GLCM). A la izquierda, la imagen original. A la derecha, la representación gráfica de la GLCM calculada para $d = 1$ y $\theta = 0^\circ$. La fuerte concentración de valores en la diagonal principal indica que la mayoría de los píxeles vecinos tienen intensidades idénticas o muy similares (alta homogeneidad), correspondiendo a las áreas uniformes de los círculos y el fondo, mientras que los valores fuera de la diagonal representan los bordes (transiciones de contraste).	20
2.6. Visualización de la respuesta de un filtro de Gabor. A la izquierda, la imagen original. A la derecha, el mapa de magnitud de la respuesta tras aplicar un filtro con una orientación y frecuencia específicas. Se observa cómo el filtro resalta enérgicamente los contornos de los círculos (bordes) suprimiendo las zonas homogéneas, lo que ilustra la capacidad del método para discriminar patrones basándose en su frecuencia espacial y direccionalidad.	21

- 2.7. Representación gráfica de un SVM lineal en un espacio bidimensional. El hiperplano óptimo (línea continua gris oscura) separa las dos clases maximizando el margen, delimitado por las líneas discontinuas. Los puntos resaltados con un círculo negro se encuentran exactamente sobre las fronteras del margen y constituyen los *vectores de soporte*, siendo los únicos datos que determinan la posición del hiperplano. . . . 24

- 2.8. Ilustración conceptual del *kernel trick* aplicado a datos no linealmente separables. A la izquierda (Espacio Original 2D), las clases forman anillos concéntricos imposibles de separar mediante una recta. A la derecha (Espacio de Características 3D), tras aplicar una función de mapeo no lineal ϕ , los datos se proyectan a una dimensión superior donde se vuelven linealmente separables mediante un hiperplano (representado por la superficie gris). 25

- 2.9. Representación esquemática de la regla de decisión del algoritmo k-Nearest Neighbors (k-NN) configurado con $k = 3$. El punto a clasificar (estrella amarilla) se evalúa calculando la distancia respecto a las muestras de entrenamiento. La decisión final se toma por votación mayoritaria entre los k vecinos más cercanos (conectados por líneas discontinuas); en este ejemplo, dado que los tres vecinos más próximos pertenecen a la Clase B (rojos), el nuevo punto es asignado a dicha clase. 27

- 2.10. Visualización de un clasificador lineal (Regresión Logística). La frontera de decisión es una recta (hiperplano en dimensiones superiores) que divide el espacio en dos regiones. Este modelo es eficiente pero incapaz de separar datos con estructuras curvas o complejas. 28

- 2.11. Frontera de decisión generada por un Random Forest sobre datos no linealmente separables. A diferencia de los modelos lineales, el ensemble de árboles puede modelar contornos complejos y regiones de decisión discontinuas, adaptándose mejor a la estructura intrínseca de los datos. 28

- 2.12. Representación de un clasificador Naïve Bayes Gaussiano. El modelo estima las distribuciones de probabilidad subyacentes de cada clase (representadas por las elipses punteadas y el gradiente de color). La decisión de clasificación se basa en la probabilidad posterior de pertenencia a cada clase. 29

2.13. Diagrama esquemático de una Red Neuronal Convolutiva (CNN). El bloque azul representa la etapa de extracción de características, donde los filtros convolucionales transforman la imagen en mapas de activación. El bloque rojo (Flatten/Embedding) representa el vector de características latente: en un entrenamiento <i>end-to-end</i> , este vector alimenta a las capas densas (naranja) para clasificar; en la metodología híbrida de este TFG, este vector se extrae para entrenar una SVM externa.	30
2.14. Flujo de transformación de datos en una arquitectura CNN básica. (1) La imagen de entrada es procesada por capas convolucionales que aplican filtros para detectar patrones locales. (2) La función de activación ReLU introduce no linealidad. (3) Las capas de pooling reducen la dimensionalidad espacial conservando la información más relevante. (4) Finalmente, los mapas de características se aplanan (<i>flatten</i>) para alimentar las capas totalmente conectadas que generan la predicción final (5).	31
2.15. Esquema de un bloque residual básico. La conexión de salto (<i>shortcut</i>) realiza una identidad que se suma a la salida de las capas de convolución $\mathcal{F}(x)$. Esta arquitectura permite mitigar el problema del desvanecimiento del gradiente.	37
2.16. Comparación arquitectónica entre el Bloque Residual Básico (izquierda) y el Bloque <i>Bottleneck</i> (derecha) utilizado en ResNet-50. Nótese la estructura de "sándwich" (1x1, 3x3, 1x1) en la derecha que optimiza el cómputo.	38
2.17. Muestra aleatoria de categorías del Describable Textures Dataset (DTD). Se muestran 12 ejemplos extraídos del conjunto de datos local, abarcando clases semánticas como <i>banded</i> (bandeado), <i>honey-combed</i> (apanalado) o <i>fibrous</i> (fibroso). Nótese la alta variabilidad intra-clase en términos de iluminación, perspectiva y escala, lo que convierte a este dataset en un desafío para los algoritmos de clasificación.	45

5.1. Curvas de aprendizaje agregadas para ResNet-50 <i>Frozen</i> . La sombra representa la desviación estándar entre <i>splits</i> . Se observa que la pérdida de validación se estabiliza tras las primeras épocas, mientras que la pérdida de entrenamiento continúa descendiendo, lo que indica que la adaptación de las características preentrenadas al dominio del DTD es limitada bajo este esquema.	60
5.2. Curvas de aprendizaje agregadas para ResNet-50 <i>Fine-Tuning</i> . A diferencia del modelo congelado, el ajuste fino muestra una mejora progresiva tanto en pérdida como en precisión. Sin embargo, la separación entre las curvas de entrenamiento y validación tras las primeras épocas refleja signos de sobreajuste, pese al incremento en precisión final.	60
5.3. Comparativa de matrices de confusión normalizadas agregadas (10 <i>splits</i>). Se observa una diagonal significativamente más definida y limpia en el modelo de aprendizaje profundo (b), frente a la dispersión de ruido en el modelo clásico (a).	61
5.4. Mapas de activación Grad-CAM en ResNet-50. Las zonas cálidas indican las regiones que maximizan la clase predicha, coincidiendo con los patrones repetitivos de la textura.	63
7.1. Mapa Grad-CAM para un ejemplo de la clase <i>cobwebbed</i> . Las zonas cálidas se alinean con la estructura de hilos entrecruzados típica de esta textura, indicando que la red fundamenta su decisión en el patrón textural relevante.	68

1. Introducción

1.1. Contexto y motivación

En las últimas décadas, la visión por computador se ha consolidado como una tecnología clave en numerosos sectores industriales, desde el control de calidad en líneas de producción hasta la inspección visual, la monitorización de procesos y la navegación de robots móviles. En todos estos ámbitos, la capacidad de analizar automáticamente el aspecto superficial de materiales, piezas o entornos resulta esencial para detectar defectos, clasificar productos o supervisar el estado de una instalación.

Dentro de este contexto, la clasificación de texturas ocupa un papel central. La textura contiene información sobre la estructura local y global de la superficie de un objeto o escena y permite distinguir entre materiales con apariencia similar, detectar anomalías en recubrimientos o pavimentos, o identificar patrones relevantes en sensores visuales empleados en tareas de automatización. Por ejemplo, la diferencia entre una superficie lisa y otra rugosa, o entre un tejido homogéneo y otro defectuoso, puede manifestarse principalmente a través de variaciones texturales que resultan difíciles de capturar mediante descriptores puramente geométricos o colorimétricos.

Tradicionalmente, estas tareas se han abordado mediante descriptores clásicos de textura, tales como Local Binary Patterns (LBP), las características derivadas de matrices de coocurrencia de niveles de gris (Haralick/GLCM) o los bancos de filtros de Gabor. Estos métodos han demostrado ser eficaces en numerosos problemas, especialmente cuando se dispone de recursos computacionales limitados y se requiere un comportamiento relativamente interpretable y estable.

En paralelo, el avance del aprendizaje profundo (Deep Learning) y, en particular, de las redes neuronales convolucionales (Convolutional Neural Networks, CNN), ha supuesto un cambio de paradigma en la forma de abordar la clasificación de imágenes, incluyendo la clasificación de texturas. Las arquitecturas convolucionales de última generación (state-of-the-art, SOTA), como las familias ResNet, han mostrado un rendimiento sobresaliente en benchmarks de visión por computador y se han convertido en la opción de referencia en muchas aplicaciones donde la precisión es prioritaria.

Sin embargo, en el ámbito de la ingeniería de automatización, la elección entre descriptores clásicos y modelos basados en CNN no puede depender exclusivamente de la precisión. Factores como la eficiencia computacional en términos de tiempo de

inferencia y requisitos hardware, y el grado de explicabilidad del modelo (es decir, la capacidad de entender qué patrones visuales sustentan una decisión) resultan determinantes a la hora de integrar estas soluciones en sistemas industriales reales.

En este contexto, resulta de especial interés disponer de un estudio comparativo sistemático que analice, bajo un mismo marco experimental, el comportamiento de descriptores clásicos y arquitecturas CNN SOTA en tareas de clasificación de texturas, considerando no solo el rendimiento, sino también la eficiencia y la explicabilidad. El presente Trabajo de Fin de Grado, titulado “Análisis Comparativo de Descriptores Clásicos y Embeddings de ResNet-50: Evaluación en texturas complejas”, se sitúa precisamente en este punto, utilizando el Describable Textures Dataset (DTD) como banco de pruebas de referencia para evaluar estas metodologías en un entorno representativo de aplicaciones de automatización.

1.2. Problema de investigación

La literatura recoge un gran número de trabajos dedicados tanto al diseño de descriptores clásicos de textura como al desarrollo de arquitecturas convolucionales cada vez más profundas y eficientes. No obstante, estos estudios suelen analizar cada familia de métodos de forma relativamente aislada o bien centran la comparación únicamente en métricas de rendimiento (por ejemplo, la tasa de acierto en un conjunto de prueba), sin abordar de manera conjunta otros aspectos igualmente relevantes para la automatización industrial.

En particular, se pueden identificar tres limitaciones frecuentes en los trabajos existentes:

- Un análisis incompleto de la eficiencia computacional, que en muchos casos se limita a mencionar el tipo de hardware utilizado, sin cuantificar de forma sistemática tiempos de inferencia, imágenes por segundo (FPS) o consumo de recursos.
- Un tratamiento limitado de la explicabilidad, especialmente en el caso de redes profundas, donde a menudo se asume la red como una “caja negra” y no se explora de forma detallada qué regiones o patrones texturales influyen más en la decisión del modelo.

Adicionalmente, muchas comparativas combinan descriptores y clasificadores de manera heterogénea (por ejemplo, distintos tipos de SVM, redes completamente conectadas o clasificadores específicos de cada arquitectura), lo que dificulta aislar hasta

qué punto las diferencias observadas se deben a la representación de características o al modelo de decisión utilizado.

Como consecuencia, un ingeniero que deba seleccionar entre un descriptor clásico (por ejemplo, LBP+SVM) y una arquitectura CNN SOTA (por ejemplo, ResNet-50 con aprendizaje transferido) carece, en muchos casos, de una guía cuantitativa y cualitativa que le permita tomar decisiones informadas en función de las restricciones de su aplicación (tiempo real, capacidad de cómputo, necesidad de interpretabilidad, etc.).

El problema de investigación que aborda este TFG puede formularse, de forma resumida, como: *¿Hasta qué punto las arquitecturas CNN SOTA superan a los descriptores clásicos de textura en términos de rendimiento y explicabilidad, y qué coste computacional conlleva esta posible mejora en el contexto de aplicaciones de automatización cuando ambos se evalúan bajo un mismo marco experimental?*

Responder a esta pregunta requiere diseñar un marco experimental que permita comparar de manera justa ambas familias de métodos, bajo las mismas condiciones de entrenamiento y evaluación, utilizando un dataset de referencia como DTD y empleando métricas que capturen los distintos aspectos de interés. En particular, el trabajo se apoya en un tipo de clasificador unificado de tipo SVM como “juez común” para evaluar vectores de características generados tanto por descriptores clásicos como por embeddings derivados de CNN, complementando este análisis con configuraciones CNN end-to-end cuando resulta pertinente.

1.3. Preguntas de investigación y enfoque del estudio

A partir del problema planteado en el apartado anterior y de los objetivos definidos en la Sección 1.4, este Trabajo de Fin de Grado se organiza como un estudio comparativo centrado en un conjunto acotado de métodos de clasificación de texturas, sin formular hipótesis estadísticas formales. El interés principal no es demostrar una afirmación cerrada, sino caracterizar empíricamente el comportamiento de tres descriptores clásicos (LBP, Haralick/GLCM y filtros de Gabor) y de una arquitectura convolucional representativa (ResNet-50) cuando se evalúan bajo un mismo marco experimental sobre el Describable Textures Dataset (DTD). En este contexto, el trabajo se articula en torno a las siguientes preguntas:

- **Rendimiento en clasificación de texturas:**

- ¿Qué diferencias de rendimiento (por ejemplo, en términos de accuracy

y F1-score macro) se observan entre los descriptores clásicos de textura (LBP, Haralick/GLCM y Gabor) combinados con una SVM y los embeddings extraídos de ResNet-50 (en configuraciones congelada y con fine-tuning) cuando todos ellos se evalúan sobre DTD bajo condiciones comparables de entrenamiento y prueba?

- ¿Cómo se sitúa el rendimiento de estos esquemas basados en “extractor de características + SVM” frente al de la propia ResNet-50 entrenada end-to-end sobre DTD con su cabeza de clasificación softmax?

■ Comportamiento frente a errores y confusiones entre clases:

- ¿Qué tipos de errores y patrones de confusión aparecen en las matrices de confusión de los distintos métodos (descriptores clásicos + SVM, embeddings de ResNet-50 + SVM y ResNet-50 end-to-end)?
- ¿Se observan diferencias claras en el modo en que cada familia de representaciones tiende a confundir determinadas categorías de textura del DTD?

■ Eficiencia computacional y viabilidad en automatización:

- ¿Qué tiempos de inferencia por imagen y qué tasas de procesamiento (imágenes por segundo, FPS) alcanzan los distintos métodos en un entorno de ejecución representativo de aplicaciones de automatización?
- ¿En qué medida el mayor coste computacional asociado a las arquitecturas CNN de última generación se justifica por las posibles mejoras en rendimiento, especialmente en escenarios con restricciones de tiempo real o recursos hardware limitados?

■ Explicabilidad y análisis de decisiones del modelo:

- ¿Qué tipo de información aportan los mapas de activación obtenidos mediante Grad-CAM aplicados a los modelos de ResNet-50 entrenados sobre DTD (especialmente la variante con fine-tuning) acerca de las regiones y patrones texturales más relevantes para la decisión del modelo?
- ¿En qué medida esta información cualitativa puede relacionarse con las propiedades que capturan los descriptores clásicos (histogramas LBP, medidas de coocurrencia, respuestas de filtros de Gabor) y ayudar a interpretar las diferencias observadas en rendimiento?

El enfoque del estudio consiste, por tanto, en diseñar y ejecutar un conjunto de experimentos que permitan responder a estas preguntas de forma coherente y reproducible. Para ello, en los capítulos posteriores se detallan:

- el marco teórico y el estado del arte en clasificación de texturas, descriptores clásicos, CNN y técnicas de explicabilidad (Capítulo 2);
- el diseño metodológico y los criterios de evaluación adoptados, incluyendo el papel de la SVM como clasificador unificado sobre descriptores clásicos y embeddings de ResNet-50 (Capítulo 3);
- el desarrollo práctico del sistema y la implementación de los distintos métodos (Capítulo 4);
- y, finalmente, la presentación, análisis y discusión de los resultados obtenidos (Capítulos 5 y 6), a partir de los cuales se formulan las conclusiones y líneas de trabajo futuro (Capítulos 7 y 8).

1.4. Objetivos

1.4.1. Objetivo general

El objetivo general de este Trabajo de Fin de Grado es: Realizar un análisis comparativo sistemático del rendimiento, la eficiencia y la explicabilidad de una arquitectura CNN SOTA (ResNet-50) frente a descriptores clásicos de textura, aplicados a la clasificación de texturas en el Describable Textures Dataset (DTD).

1.4.2. Objetivos específicos

Para alcanzar este objetivo general se definen los siguientes objetivos específicos:

1. Revisar el estado del arte en clasificación de texturas, abarcando tanto descriptores clásicos (LBP, Haralick/GLCM, filtros de Gabor) como arquitecturas convolucionales modernas y técnicas de explicabilidad aplicadas a visión por computador.
2. Diseñar un marco experimental común que permita comparar, bajo condiciones homogéneas, distintas configuraciones de:
 - descriptores clásicos de textura combinados con clasificadores tradicionales, empleando principalmente un clasificador de tipo SVM como referencia unificada;

- una arquitectura CNN SOTA preentrenada (ResNet-50) mediante aprendizaje transferido, tanto en modo end-to-end como extractores de características alimentando a la misma SVM.
3. Evaluar el rendimiento en clasificación de texturas de los métodos considerados, utilizando DTD como dataset de referencia y métricas estándar (accuracy, F1-score, matrices de confusión, etc.).
 4. Medir la eficiencia computacional de los métodos evaluados, estimando tiempos de inferencia por imagen, tasas de procesamiento (FPS) y, cuando sea posible, el impacto en el uso de recursos hardware, para valorar su viabilidad en sistemas de automatización con restricciones de tiempo real.
 5. Aplicar técnicas de explicabilidad (como Grad-CAM u otras herramientas de XAI visual) a las arquitecturas basadas en CNN, con el fin de analizar qué regiones de las texturas resultan más relevantes para la decisión del modelo y comparar este comportamiento con la información aportada por los descriptores clásicos.
 6. Sintetizar y discutir los resultados, identificando las principales ventajas, limitaciones y trade-offs entre descriptores clásicos y arquitecturas CNN SOTA, y formulando recomendaciones orientadas a su uso en aplicaciones de automatización basadas en visión de texturas.

1.5. Alcance y limitaciones

El presente Trabajo de Fin de Grado se centra en el análisis comparativo de distintas familias de métodos para la clasificación de texturas desde una perspectiva orientada a aplicaciones de automatización. Para acotar el problema y garantizar la viabilidad del estudio, se establecen las siguientes consideraciones de alcance:

- El trabajo se limita a la clasificación de texturas a nivel de imagen. No se abordan otras tareas relacionadas, como la segmentación de texturas, la detección de objetos o el análisis temporal de secuencias de vídeo.
- El estudio se desarrolla sobre un único dataset de referencia, el Describable Textures Dataset (DTD), por ser un conjunto de datos ampliamente utilizado en la literatura para evaluar métodos de clasificación de texturas. No se incluyen otros datasets, por lo que la generalización de los resultados a dominios distintos debe considerarse con cautela.

- En el ámbito de los descriptores clásicos, se analizan de forma específica Local Binary Patterns (LBP), características derivadas de matrices de coocurrencia (Haralick/GLCM) y filtros de Gabor, combinados principalmente con un clasificador de tipo SVM. Existen otros descriptores en la literatura que no se consideran en este trabajo.
- En el ámbito del aprendizaje profundo, se evalúa con una arquitectura preentrenada de última generación (ResNet-50) empleando técnicas de aprendizaje transferido y, en su caso, ajuste fino. El análisis no pretende cubrir de forma exhaustiva todas las arquitecturas CNN disponibles.

En cuanto a las limitaciones principales del trabajo, pueden destacarse las siguientes:

- Los resultados obtenidos dependen de las configuraciones concretas empleadas (hiperparámetros, esquemas de entrenamiento, criterios de parada, etc.). Aunque se busca un ajuste razonable de cada método, no se pretende realizar una optimización exhaustiva del espacio de hiperparámetros.
- Las evaluaciones de eficiencia computacional (tiempo de inferencia, imágenes por segundo) se realizan en un entorno hardware específico, representativo pero limitado. Por tanto, los valores numéricos no son directamente extrapolables a otras plataformas con diferentes capacidades de cómputo.
- El análisis de explicabilidad se basa principalmente en técnicas de visualización de mapas de activación, como Grad-CAM u otras metodologías afines. Estas herramientas proporcionan información útil, pero no agotan el conjunto de enfoques disponibles en el campo de la inteligencia artificial explicable.
- El estudio no incluye una validación en un entorno industrial real, sino que se realiza en un escenario de laboratorio utilizando un dataset público. La transferencia de los resultados a aplicaciones concretas de automatización requeriría considerar aspectos adicionales como la variabilidad real de las condiciones de captura, el mantenimiento del sistema o su integración con otros componentes de control.

En conjunto, el trabajo proporciona una comparación detallada y coherente dentro del marco definido, pero sus conclusiones deben interpretarse teniendo en cuenta estas restricciones de alcance y las condiciones experimentales bajo las cuales se han obtenido los resultados.

1.6. Estructura de la memoria

La memoria se organiza en ocho capítulos, además de las referencias bibliográficas y, en su caso, los anexos, con el fin de presentar el trabajo de forma ordenada y coherente.

- En el **Capítulo 1 – Introducción** se presenta el contexto y la motivación del trabajo, se formula el problema de investigación, se plantean las preguntas que guían el estudio, se definen los objetivos generales y específicos y se delimita el alcance y las principales limitaciones del proyecto.
- El **Capítulo 2 – Fundamentos teóricos y estado del arte** recoge los conceptos básicos y los trabajos relacionados necesarios para enmarcar el estudio. Se revisan la clasificación de texturas en visión por computador, los descriptores clásicos de textura considerados (LBP, Haralick/GLCM y filtros de Gabor), los clasificadores tradicionales, los fundamentos de las redes neuronales convolucionales y las arquitecturas SOTA empleadas. Asimismo, se introducen las técnicas de aprendizaje transferido, las metodologías de explicabilidad en visión (como Grad-CAM) y los aspectos de eficiencia en modelos de visión, junto con una descripción del dataset DTD.
- En el **Capítulo 3 – Metodología y diseño experimental** se describe el planteamiento general del estudio comparativo. Se detallan los métodos evaluados, la preparación de los datos, el diseño de los distintos escenarios de evaluación (rendimiento, eficiencia y explicabilidad) y las métricas utilizadas, así como el procedimiento experimental seguido para garantizar la reproducibilidad de los resultados.
- El **Capítulo 4 – Desarrollo experimental** se centra en la implementación práctica del sistema. Se describen el entorno de desarrollo y hardware utilizado, la estructura del pipeline de descriptores clásicos, la configuración de ResNet-50 como arquitectura CNN de referencia en modo end-to-end y como extractor de características, así como la organización de los módulos destinados a la evaluación de eficiencia y explicabilidad.
- El **Capítulo 5 – Resultados** presenta los resultados obtenidos en los diferentes escenarios de evaluación. Se muestran y analizan las métricas de rendimiento en clasificación de texturas, las medidas de eficiencia computacional y los ejemplos de mapas de explicabilidad generados para las arquitecturas basadas en CNN.

- En el **Capítulo 6 – Discusión** se interpretan los resultados de forma conjunta, comparando el comportamiento de los descriptores clásicos y de la arquitectura ResNet-50. Se analizan las implicaciones de estos resultados para aplicaciones de automatización, se contrastan con trabajos previos y se discuten las principales limitaciones y posibles fuentes de sesgo del estudio.
- El **Capítulo 7 – Conclusiones** resume las aportaciones principales del trabajo y responde a las preguntas de investigación planteadas en la introducción, poniendo de relieve las implicaciones prácticas del análisis comparativo realizado.
- Finalmente, el **Capítulo 8 – Trabajo futuro** propone posibles líneas de continuidad, tanto en términos de ampliación metodológica (nuevos descriptores, arquitecturas o métricas) como de aplicación en entornos industriales reales o en otros dominios de visión por computador.

Tras estos capítulos se incluyen las referencias bibliográficas, recopiladas siguiendo el formato establecido por la Escuela, y, en su caso, anexos con información complementaria, como configuraciones detalladas de entrenamiento, tablas ampliadas de resultados o ejemplos adicionales de visualizaciones de explicabilidad.

2. Fundamentos teóricos y estado del arte

2.1. Visión por computador y clasificación de texturas

La visión por computador estudia métodos automáticos para extraer información útil a partir de imágenes o secuencias de vídeo, con aplicaciones que abarcan desde la percepción para robots hasta el control de calidad, la inspección visual o la monitorización de procesos industriales. En este contexto, la textura constituye una de las propiedades visuales fundamentales, junto con la forma, el color o la intensidad, y recoge patrones de variación espacial que caracterizan la apariencia de superficies y materiales.



Figura 2.1: Flujo habitual de un sistema de visión por computador

Desde un punto de vista clásico, la textura puede entenderse como la organización espacial de elementos básicos (primitivas) que se repiten de forma más o menos regular, generando una sensación de rugosidad, granulado, direccionalidad o uniformidad. En la práctica, la textura se manifiesta como distribuciones características de intensidades, colores o gradientes en la imagen y resulta especialmente relevante cuando las diferencias entre clases no pueden describirse únicamente mediante contornos bien definidos o cambios bruscos de color.

La clasificación de texturas se formula habitualmente como un problema de reconocimiento de patrones: a partir de una imagen (o un parche de imagen) se extraen características que describen su textura y, a continuación, se asigna esa muestra a una de las clases predefinidas mediante un clasificador. La calidad del sistema depende, por tanto, tanto del método de representación (descriptores de textura) como del modelo de decisión (clasificador). La literatura distingue varias familias de méto-

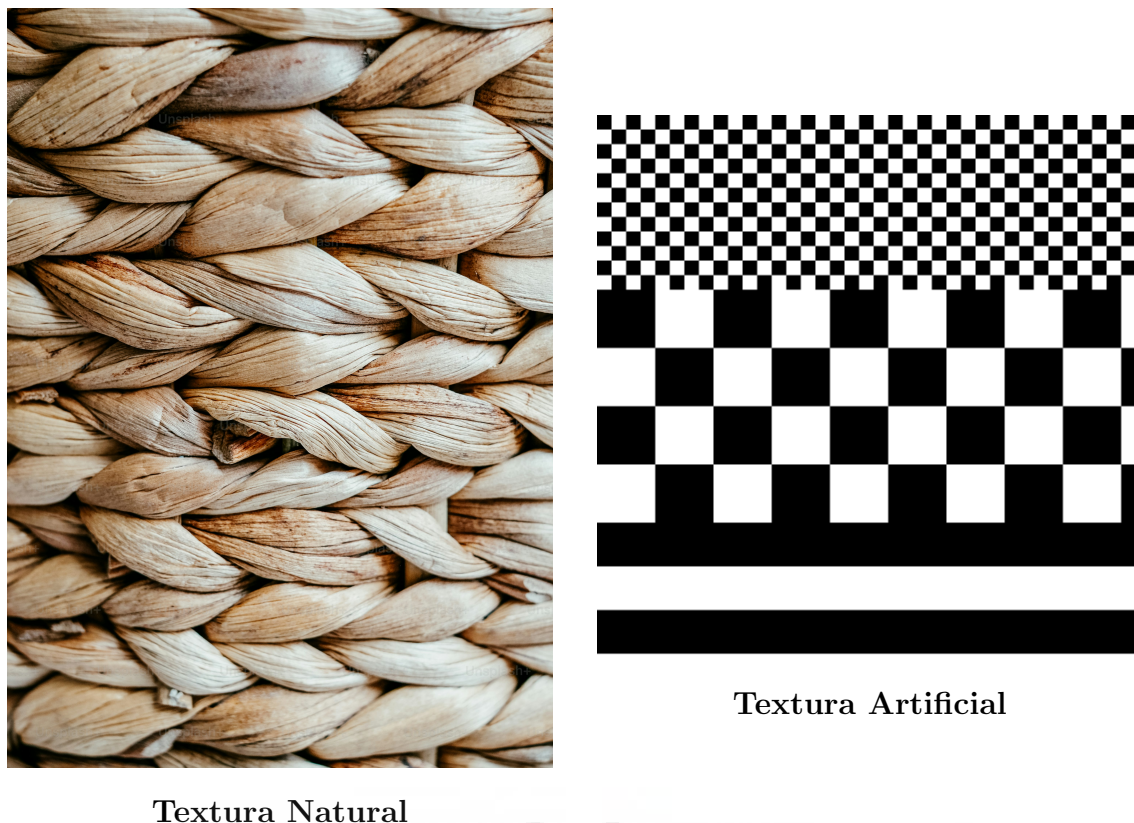


Figura 2.2: Comparativa visual: Textura natural recortada (izq) frente a patrón artificial (der).

dos de análisis de textura, que incluyen aproximaciones estadísticas, estructurales, basadas en modelos y basadas en transformadas.

En los últimos años, el desarrollo del aprendizaje profundo ha modificado de manera significativa este panorama. Las redes neuronales convolucionales (Convolutional Neural Networks, CNN) han demostrado una capacidad notable para aprender representaciones jerárquicas directamente a partir de datos, alcanzando resultados de referencia en tareas de reconocimiento de objetos y clasificación de imágenes en grandes bases de datos como ImageNet. Este avance también ha tenido impacto en la representación de texturas, donde las CNN y los descriptores derivados de activaciones intermedias han ido sustituyendo progresivamente a muchos descriptores manuales, especialmente en escenarios con grandes volúmenes de datos y recursos de cómputo suficientes.

No obstante, en aplicaciones de automatización, la elección de la técnica de clasificación de texturas no depende únicamente de la precisión. Factores como la facilidad de implementación, el coste computacional y la explicabilidad del modelo son determinantes para su adopción en entornos industriales. En este contexto, los descriptores clásicos continúan siendo una alternativa válida cuando se requiere simplicidad, efi-

ciencia y un comportamiento razonablemente interpretable, mientras que las CNN ofrecen mayor flexibilidad y potencial de rendimiento a costa de una mayor complejidad.

El presente trabajo se sitúa precisamente en esta intersección entre visión por computador, clasificación de texturas y ingeniería de automatización. Como banco de pruebas se adopta el Describable Textures Dataset (DTD), un conjunto de datos de texturas “en el mundo real” que agrupa miles de imágenes en 47 categorías descritas mediante atributos semánticos como *banded*, *striped* o *wrinkled*, lo que permite estudiar representaciones de textura tanto desde una perspectiva tradicional como mediante técnicas modernas de aprendizaje profundo. En este marco, el TFG se centra en comparar diferentes representaciones de textura, clásicas y basadas en CNN, bajo un mismo esquema de clasificación, utilizando principalmente una SVM como clasificador unificado, de acuerdo con el enfoque definido en el Capítulo 1.

2.1.1. Concepto de textura y propiedades relevantes

La definición de “textura” no es única, pero la mayoría de los autores coinciden en que se trata de una propiedad asociada a la distribución espacial de las intensidades o colores en una imagen, que refleja la estructura interna de la superficie representada. Frente a características locales puramente geométricas, la textura captura patrones relacionados con:

- la homogeneidad o variabilidad local (regiones lisas frente a regiones rugosas),
- la presencia de direccionalidad (texturas estriadas, alineadas),
- la existencia de periodicidad o patrones repetitivos,
- y la combinación de estructuras a diferentes escalas.

En la literatura se han propuesto diversas taxonomías para los métodos de análisis de textura. Una clasificación habitual distingue cuatro grandes categorías:

- **Métodos estadísticos**, que describen la textura a partir de medidas sobre la distribución de intensidades o relaciones de coocurrencia entre píxeles (por ejemplo, histogramas, matrices de coocurrencia, momentos estadísticos).
- **Métodos estructurales**, que modelan la textura como la combinación de primitivas básicas (por ejemplo, segmentos, puntos, regiones) dispuestas según reglas de colocación más o menos regulares.

- **Métodos basados en modelos**, que suponen que la textura se genera mediante un proceso probabilístico o físico subyacente (por ejemplo, modelos de campos aleatorios de Markov).
- **Métodos basados en transformadas**, que analizan la energía o respuesta de la imagen en distintos dominios (por ejemplo, transformadas de Gabor, wavelets o representaciones en frecuencia).

En la práctica, muchos descriptores concretos combinan ideas de varias categorías. Por ejemplo, un banco de filtros de Gabor genera una representación en el dominio de la frecuencia y la orientación, pero sus respuestas se resumen a menudo mediante estadísticas simples (energía media, varianza) en cada canal.

En el ámbito de este TFG, la textura se considera una propiedad intrínseca de la imagen que puede describirse de forma efectiva mediante:

- patrones locales de intensidad (como en LBP),
- distribuciones de coocurrencia (como en las características de Haralick),
- o respuestas a filtros sintonizados en frecuencia y orientación (como en los filtros de Gabor).

Estos descriptores generan vectores de características de dimensión fija que pueden utilizarse como entrada para clasificadores supervisados, en particular para la SVM que se adopta en este trabajo como clasificador de referencia. La formalización detallada de estos descriptores se desarrolla en la Sección 2.2.

2.1.2. Aplicaciones de la clasificación de texturas en automatización

La clasificación de texturas tiene un papel destacado en numerosas aplicaciones de ingeniería y automatización, donde la superficie del material o del entorno aporta información crítica sobre el estado del sistema. Algunos ejemplos representativos son:

- **Control de calidad en fabricación:** detección de defectos en superficies metálicas, cerámicas o textiles, donde pequeñas variaciones en la textura pueden indicar rayaduras, poros, grietas o irregularidades en el proceso de producción.

- **Inspección de recubrimientos y pavimentos:** evaluación del desgaste de suelos industriales, carreteras o pistas de aeropuertos, así como control del acabado de pinturas y recubrimientos protectores.
- **Clasificación de materiales:** identificación automática de tipos de madera, tejidos, composites o materiales granulados, útil en líneas de clasificación y reciclaje.
- **Monitorización de procesos:** análisis de la apariencia de espumas, mezclas o emulsiones en procesos químicos y alimentarios, donde la textura puede correlacionarse con propiedades físicas (viscosidad, homogeneidad, presencia de burbujas).
- **Percepción para robótica:** reconocimiento de entornos y superficies (por ejemplo, suelos lisos frente a terrenos rugosos) que condicionan la planificación del movimiento y la interacción del robot con el entorno.

En estos escenarios, la viabilidad práctica de la técnica de clasificación de texturas está condicionada por factores como:

- la disponibilidad de datos representativos del proceso (número y diversidad de muestras),
- las restricciones de tiempo real (tiempo máximo permitido por imagen),
- la capacidad de cómputo del hardware embebido o del sistema de visión,
- y el grado de explicabilidad exigido para su integración en sistemas críticos o regulados.

Los descriptores clásicos han sido ampliamente utilizados en sistemas de inspección industrial debido a su relativa sencillez, bajo coste computacional y facilidad de implementación, especialmente cuando se integran en cámaras inteligentes o PLCs con recursos limitados. Por su parte, las CNN permiten capturar patrones de textura complejos y no lineales, pero su adopción en entornos de automatización exige evaluar cuidadosamente el equilibrio entre ganancia de rendimiento y coste computacional, así como el uso de técnicas de explicabilidad que faciliten la validación por parte de ingenieros y especialistas.

Este contexto justifica el enfoque del presente TFG, que busca comparar de forma rigurosa ambas familias de métodos en un escenario de clasificación de texturas representativo, utilizando un marco experimental unificado (SVM como clasificador

común sobre distintos descriptores y embeddings) para ofrecer recomendaciones fundamentadas de cara a su uso en sistemas de automatización basados en visión por computador.

2.2. Descriptores clásicos de textura

Los descriptores clásicos de textura constituyen un conjunto de técnicas diseñadas para capturar, mediante reglas explícitas, los patrones de variación espacial presentes en una imagen. Frente a las CNN, que aprenden automáticamente representaciones jerárquicas a partir de datos, estos descriptores se basan en modelos manuales que reflejan hipótesis específicas sobre cómo se manifiesta la textura en el dominio de la intensidad, la coocurrencia o la frecuencia.

En el marco de este TFG se consideran tres familias de descriptores ampliamente utilizadas y representativas de diferentes enfoques:

- **Local Binary Patterns (LBP) y variantes**, que describen patrones locales de intensidad mediante codificaciones binarias invariantes a cambios de iluminación monotónicos.
- **Características derivadas de matrices de coocurrencia de niveles de gris (Haralick/GLCM)**, que capturan relaciones estadísticas de segundo orden entre pares de píxeles en distintas direcciones y distancias.
- **Filtros de Gabor**, que analizan la respuesta de la imagen en distintas frecuencias y orientaciones, proporcionando una descomposición conjunta en espacio y frecuencia relacionada con la percepción humana de la textura.

Estos métodos comparten varias propiedades relevantes para aplicaciones de automatización:

- producen vectores de características de dimensión fija, adecuados para ser utilizados con clasificadores como SVM;
- ofrecen un comportamiento relativamente interpretable, ya que muchas de sus medidas (histogramas, momentos, energías) pueden relacionarse con conceptos como rugosidad, direccionalidad o periodicidad;
- tienen un coste computacional moderado, lo que facilita su implementación en sistemas embebidos o hardware con recursos limitados.

Sin embargo, también presentan limitaciones, especialmente cuando se les compara con arquitecturas profundas modernas: su capacidad para modelar patrones muy complejos o de alto nivel es limitada, suelen requerir un diseño cuidadoso de parámetros (por ejemplo, radios y número de vecinos en LBP, desplazamientos y ángulos en GLCM, frecuencias y orientaciones en Gabor), y su rendimiento puede degradarse ante variaciones significativas no contempladas en el diseño del descriptor (por ejemplo, fuertes cambios de escala o deformaciones no rígidas).

En las subsecciones siguientes se describen con más detalle los tres descriptores clásicos considerados, poniendo el énfasis en su formulación conceptual, sus parámetros más importantes y sus ventajas y limitaciones en el contexto del presente estudio.

2.2.1. Local Binary Patterns (LBP) y variantes

El descriptor Local Binary Patterns (LBP) fue propuesto originalmente como un método sencillo y eficiente para describir patrones locales de textura mediante comparaciones de intensidad entre un píxel y su vecindario. La idea básica consiste en, para cada píxel, comparar su valor de intensidad con el de un conjunto de píxeles vecinos dispuestos sobre una circunferencia de radio fijo. Si la intensidad del vecino es mayor o igual que la del píxel central, se asigna un 1; en caso contrario, se asigna un 0. La concatenación de estas bits da lugar a un código binario que caracteriza el patrón local de intensidad.

$$LBP_{P,R} = \sum_{p=0}^{P-1} s(g_p - g_c)2^p, \quad \text{donde } s(x) = \begin{cases} 1 & \text{si } x \geq 0 \\ 0 & \text{si } x < 0 \end{cases} \quad (2.1)$$

Esta formulación garantiza que el descriptor sea invariante a cambios monotónicos de la escala de grises, ya que solo depende del signo de la diferencia de intensidades.

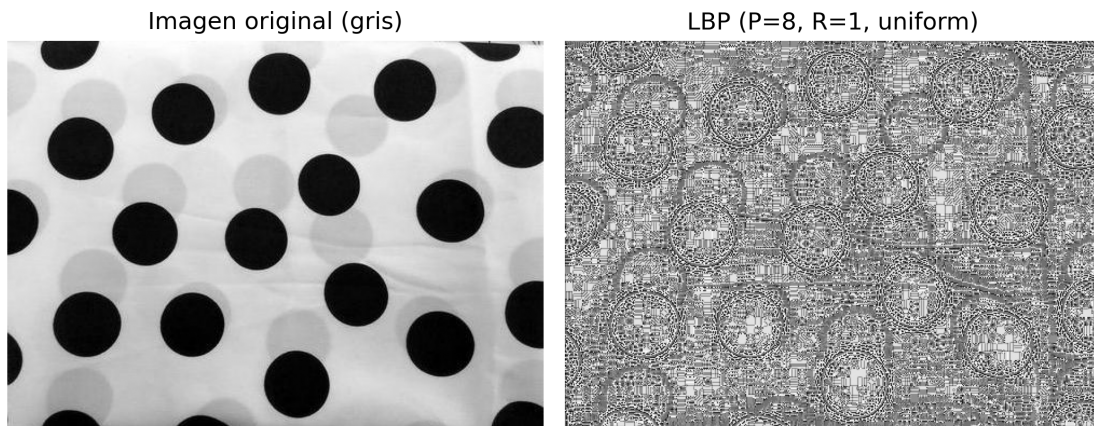


Figura 2.3: Visualización de la transformación LBP. A la izquierda, la textura original con patrones circulares oscuros. A la derecha, el mapa de códigos LBP resultante ($P = 8$, $R = 1$, uniforme), donde se aprecia cómo el descriptor resalta los bordes y la estructura local de la textura ignorando la información de color global.

Los códigos LBP obtenidos en todos los píxeles de la imagen se agrupan habitualmente en un histograma de frecuencias, que constituye el vector de características final. Este histograma resume la distribución de patrones locales de textura presentes en la imagen. Variantes posteriores del método introducen mejoras como:

- el uso de vecindarios circulares de tamaño arbitrario y muestreo bilineal para ganar invariancia a la resolución (Figura 2.3);
- la definición de patrones “uniformes”, que agrupan códigos con pocos cambios 0-1 para reducir la dimensionalidad y aumentar la robustez;
- extensiones para incorporar información de color o de varios radios de forma conjunta.

Entre las ventajas de LBP destacan su baja complejidad computacional, su relativa invariancia a cambios monotónicos de iluminación (al basarse en comparaciones de intensidad) y su buen rendimiento en múltiples tareas de clasificación de texturas y reconocimiento facial en entornos controlados. No obstante, LBP es sensible al ruido en imágenes muy degradadas y su comportamiento puede degradarse ante cambios de escala significativos o transformaciones geométricas complejas, a menos que se extienda explícitamente para abordar estos factores.

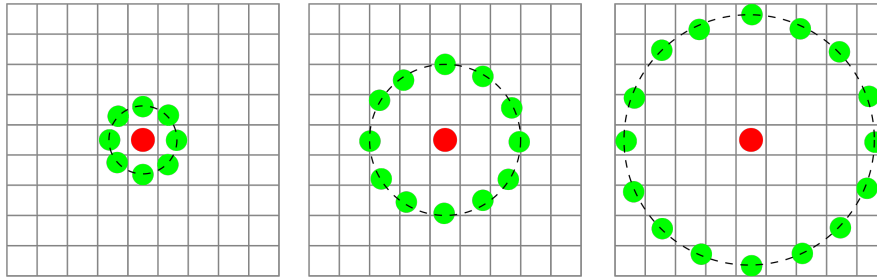


Figura 2.4: Representación de vecindarios circulares LBP extendidos para diferentes resoluciones (P, R) . De izquierda a derecha, se muestran ejemplos variando el número de puntos de muestreo (P) y el radio (R), utilizando interpolación bilineal para los puntos que no coinciden con la rejilla.

En el contexto de este TFG, LBP se utiliza como ejemplo representativo de descriptor local de textura de bajo nivel. A partir de los histogramas LBP obtenidos para cada imagen, se construyen vectores de características de dimensión fija que se suministran a la SVM empleada como clasificador unificado, lo que permite comparar su rendimiento con el de otros descriptores clásicos y con embeddings derivados de CNN bajo un mismo modelo de decisión.

2.2.2. Características Haralick y matrices GLCM

Las matrices de coocurrencia de niveles de gris (Gray-Level Co-occurrence Matrices, GLCM) constituyen una familia de descriptores estadísticos que capturan relaciones de segundo orden entre pares de píxeles a determinadas distancias y direcciones. Para una imagen cuantificada en un número finito de niveles de gris, la GLCM se define como una matriz en la que cada entrada $P(i, j)$ indica la frecuencia con la que aparece un par de intensidades (i, j) en píxeles separados por un desplazamiento $(\Delta x, \Delta y)$ determinado.

A partir de la matriz normalizada $p(i, j)$, Haralick propuso un conjunto de escalares para resumir el contenido textural. Las medidas más relevantes utilizadas en este trabajo se definen como:

$$\begin{aligned}
 \text{Contraste} &= \sum_{i,j} |i - j|^2 p(i, j) \\
 \text{Homogeneidad} &= \sum_{i,j} \frac{p(i, j)}{1 + |i - j|} \\
 \text{Energía (ASM)} &= \sum_{i,j} p(i, j)^2 \\
 \text{Entropía} &= - \sum_{i,j} p(i, j) \log(p(i, j))
 \end{aligned} \tag{2.2}$$

Estas medidas resumen distintas propiedades de la textura:

- el contraste refleja la presencia de variaciones intensas de intensidad, asociadas a texturas rugosas o con bordes marcados;
- la homogeneidad es alta en texturas uniformes, donde predominan pares de intensidades similares;
- la energía (o angular second moment) mide el grado de orden o regularidad en la distribución de coocurrencias;
- la entropía captura el grado de desorden o complejidad de la textura.

En la práctica, se suelen calcular GLCM para varios ángulos (por ejemplo, 0° , 45° , 90° y 135°) y distintas distancias, agregando posteriormente las medidas obtenidas (mediante promedios o concatenación) para construir el vector de características final.

Las características de Haralick presentan varias ventajas para aplicaciones de automatización:

- ofrecen una interpretación relativamente directa en términos de propiedades texturales (rugosidad, homogeneidad, complejidad);
- permiten capturar información estructural más allá de las distribuciones marginales de intensidad;
- pueden adaptarse a distintas resoluciones mediante la elección de distancias y cuantificación de niveles de gris.

Sin embargo, también tienen limitaciones: el tamaño de la GLCM crece con el número de niveles de gris, lo que puede requerir una cuantificación previa; la sensibilidad a



Figura 2.5: Visualización de una Matriz de Coocurrencia (GLCM). A la izquierda, la imagen original. A la derecha, la representación gráfica de la GLCM calculada para $d = 1$ y $\theta = 0^\circ$. La fuerte concentración de valores en la diagonal principal indica que la mayoría de los píxeles vecinos tienen intensidades idénticas o muy similares (alta homogeneidad), correspondiendo a las áreas uniformes de los círculos y el fondo, mientras que los valores fuera de la diagonal representan los bordes (transiciones de contraste).

rotaciones puede ser significativa si no se promedian adecuadamente las direcciones; y la selección de distancias y ángulos introduce hiperparámetros que influyen en el rendimiento.

En este TFG, las características de Haralick se emplean como ejemplo de descriptor estadístico de segundo orden. Al igual que en el caso de LBP, las medidas derivadas de las GLCM se organizan en vectores de características de dimensión fija que alimentan al clasificador SVM, facilitando una comparación directa con otros descriptores clásicos y con representaciones extraídas mediante CNN.

2.2.3. Filtros de Gabor y bancos multi-escala/multi-orientación

Los filtros de Gabor son funciones obtenidas como la modulación de una onda sinusoidal por una ventana gaussiana, lo que da lugar a kernels localizados simultáneamente en espacio y frecuencia. Esta propiedad los hace especialmente adecuados para analizar patrones de textura, ya que permiten capturar información sobre frecuencia espacial, orientación y localización de los detalles presentes en la imagen, de forma análoga a ciertos mecanismos de la percepción visual humana. Un filtro de Gabor 2D se define como una onda sinusoidal compleja modulada por una envolvente

gaussiana. Su respuesta al impulso viene dada por:

$$g(x, y; \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x'^2 + \gamma^2 y'^2}{2\sigma^2}\right) \exp\left(i\left(2\pi\frac{x'}{\lambda} + \psi\right)\right) \quad (2.3)$$

donde $x' = x \cos \theta + y \sin \theta$ y $y' = -x \sin \theta + y \cos \theta$. Los parámetros determinan la longitud de onda (λ), la orientación (θ), la fase (ψ), la desviación estándar de la gaussiana (σ) y la relación de aspecto espacial (γ).

En el análisis de texturas, es habitual emplear bancos de filtros de Gabor formados por filtros con distintas frecuencias centrales y orientaciones. Para cada filtro, se calcula la respuesta de convolución sobre la imagen y se extraen estadísticas simples (por ejemplo, energía media, desviación estándar) de la magnitud de la respuesta en cada canal. La concatenación de estas estadísticas para todas las combinaciones de frecuencia y orientación genera un vector de características que describe la textura en términos de su contenido direccional y frecuencial.

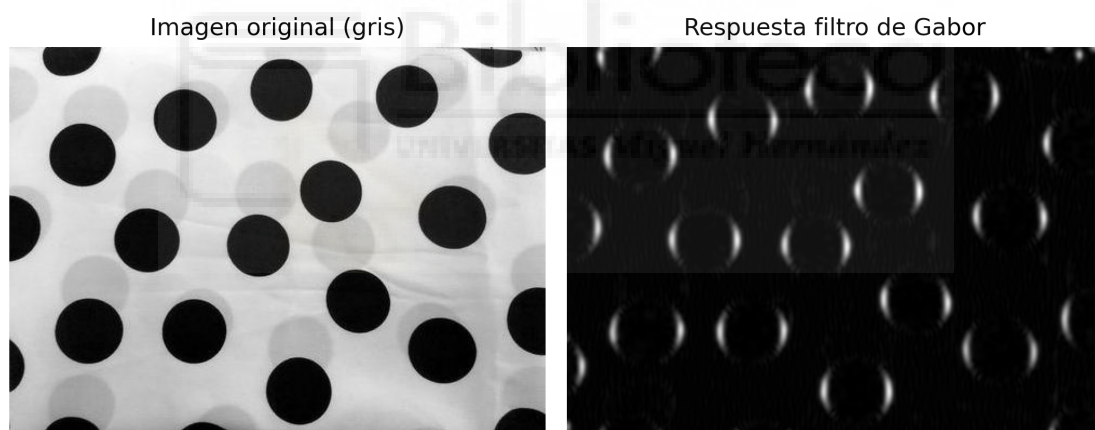


Figura 2.6: Visualización de la respuesta de un filtro de Gabor. A la izquierda, la imagen original. A la derecha, el mapa de magnitud de la respuesta tras aplicar un filtro con una orientación y frecuencia específicas. Se observa cómo el filtro resalta energéticamente los contornos de los círculos (bordes) suprimiendo las zonas homogéneas, lo que ilustra la capacidad del método para discriminar patrones basándose en su frecuencia espacial y direccionalidad.

Entre las ventajas de los filtros de Gabor destacan:

- su capacidad para capturar patrones orientados y periódicos, como estrías, bandas o retículas;

- su interpretación intuitiva como filtros pasa-banda orientados, que facilitan el análisis de la textura a distintas escalas;
- su buena performance histórica en tareas de clasificación de texturas y reconocimiento de patrones estructurados.

No obstante, los bancos de Gabor presentan un coste computacional más elevado que otros descriptores clásicos, debido al número de convoluciones necesarias, y requieren decidir a priori el conjunto de frecuencias y orientaciones, lo que introduce un compromiso entre resolución, dimensionalidad del descriptor y coste de cálculo. Además, su robustez frente a grandes cambios de escala o deformaciones no rígidas puede verse limitada si el banco no está adecuadamente diseñado para cubrir dichas variaciones.

En el contexto del presente TFG, los bancos de filtros de Gabor se emplean como representante de métodos basados en transformadas y análisis frecuencial de la textura. A partir de las energías o estadísticas de respuesta de cada canal se construyen vectores de características de dimensión fija que se suministran a la SVM utilizada como clasificador unificado. De este modo, los descriptores basados en Gabor se integran en el mismo marco comparativo que LBP, Haralick y los embeddings derivados de CNN.

2.3. Clasificadores tradicionales para descriptores de textura

En un sistema de clasificación de texturas, el clasificador es el componente encargado de asignar una etiqueta de clase a cada muestra a partir del vector de características generado por el descriptor. La elección del clasificador influye de manera decisiva en el rendimiento global del sistema, especialmente cuando los descriptores producen vectores de alta dimensión o cuando el número de muestras de entrenamiento es limitado.

En la literatura se han empleado numerosas familias de clasificadores para trabajar con descriptores clásicos de textura, incluyendo métodos lineales (como la regresión logística), clasificadores basados en distancias (como k-nearest neighbors, k-NN), modelos probabilísticos (como Naïve Bayes) y técnicas de ensemble (como Random Forest). Sin embargo, una de las opciones más extendidas y exitosas, tanto en visión por computador en general como en clasificación de texturas en particular, son las Máquinas de Vectores de Soporte (Support Vector Machines, SVM).

En este Trabajo de Fin de Grado, la SVM desempeña un papel central como clasificador unificado: se utiliza como “juez común” para evaluar tanto los descriptores clásicos (LBP, Haralick, Gabor) como los embeddings derivados del ResNet-50. De este modo, las diferencias observadas en rendimiento o eficiencia se pueden atribuir principalmente a la calidad de la representación de características, reduciendo en lo posible el efecto de variaciones en el modelo de decisión.

2.3.1. Máquinas de Vectores de Soporte (SVM)

Las Máquinas de Vectores de Soporte fueron introducidas originalmente como un método de clasificación binaria basado en la búsqueda de un hiperplano que separa dos clases con el máximo margen posible. En su formulación lineal más simple, la SVM aprende un hiperplano en el espacio de características tal que:

- separa correctamente, en la medida de lo posible, las muestras de cada clase,
- y maximiza la distancia (margen) entre las muestras más cercanas de ambas clases y dicho hiperplano.

Dado un conjunto de datos de entrenamiento $\{(\mathbf{x}_i, y_i)\}$, donde $y_i \in \{-1, 1\}$, la SVM resuelve el siguiente problema de optimización cuadrática:

$$\min_{\mathbf{w}, b} \frac{1}{2} \|\mathbf{w}\|^2 \quad \text{sueto a} \quad y_i(\mathbf{w}^T \mathbf{x}_i + b) \geq 1, \quad \forall i \quad (2.4)$$

Las muestras que determinan el hiperplano óptimo se denominan *vectores de soporte*, y constituyen un subconjunto de los datos de entrenamiento que concentra la información relevante para la clasificación.

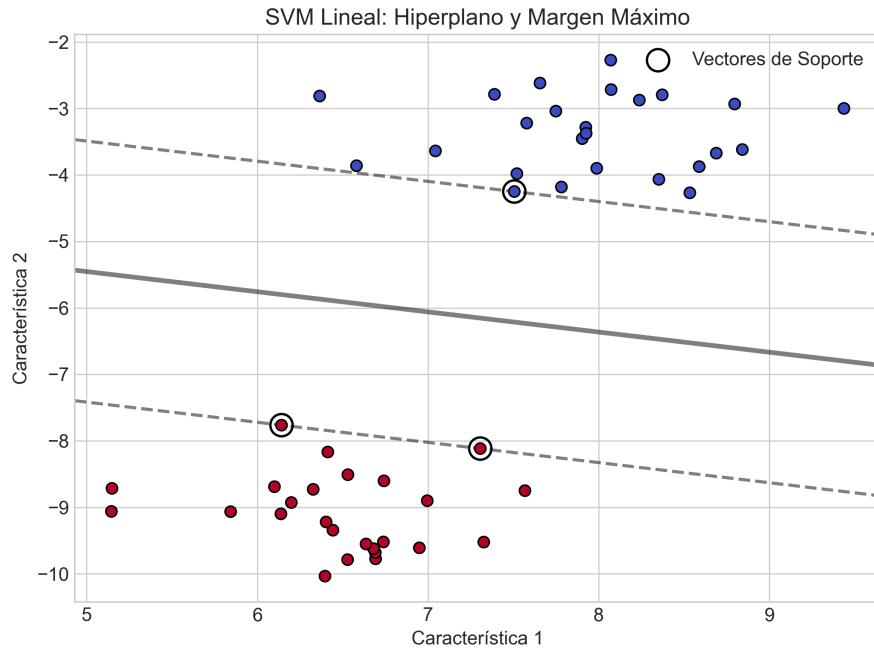


Figura 2.7: Representación gráfica de un SVM lineal en un espacio bidimensional. El hiperplano óptimo (línea continua gris oscura) separa las dos clases maximizando el margen, delimitado por las líneas discontinuas. Los puntos resaltados con un círculo negro se encuentran exactamente sobre las fronteras del margen y constituyen los *vectores de soporte*, siendo los únicos datos que determinan la posición del hiperplano.

En la práctica, la mayoría de problemas de visión por computador no son linealmente separables en el espacio de características original. Para abordar esta situación, las SVM introducen el concepto de margen suave (soft margin), que permite ciertas violaciones controladas de la separación a cambio de un término de penalización regulado por un hiperparámetro de regularización. Además, se recurre al denominado truco del kernel (kernel trick), que consiste en utilizar funciones núcleo que definen productos escalares en espacios de características de dimensión (posiblemente) muy alta sin necesidad de operar explícitamente en dichos espacios.

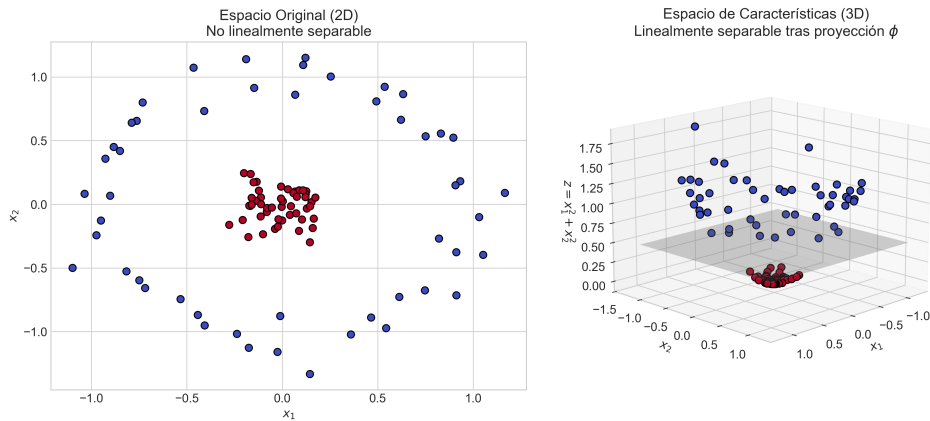


Figura 2.8: Ilustración conceptual del *kernel trick* aplicado a datos no linealmente separables. A la izquierda (Espacio Original 2D), las clases forman anillos concéntricos imposibles de separar mediante una recta. A la derecha (Espacio de Características 3D), tras aplicar una función de mapeo no lineal ϕ , los datos se proyectan a una dimensión superior donde se vuelven linealmente separables mediante un hiperplano (representado por la superficie gris).

Entre los núcleos más utilizados destacan:

- el kernel lineal, adecuado cuando las características ya incorporan relaciones no lineales y se busca un modelo simple y eficiente;
- el kernel polinómico, que introduce relaciones de orden superior entre las características;
- y el kernel radial de base gaussiana (RBF), que permite modelar fronteras de decisión no lineales y es especialmente popular en problemas de clasificación general.

La elección concreta del tipo de kernel y sus parámetros asociados (por ejemplo, el grado en el kernel polinómico o el parámetro de anchura en el kernel RBF), así como del parámetro de regularización de la SVM, se realiza habitualmente mediante procedimientos de validación cruzada o búsqueda sistemática en rejilla. En el contexto de este TFG, la configuración de la SVM se detalla en el Capítulo 3, donde se describe el esquema de validación y ajuste de hiperparámetros adoptado para garantizar una comparación justa entre métodos.

Aunque la formulación original de la SVM es binaria, la clasificación de texturas en el DTD requiere abordar un problema multiclase con decenas de categorías. Para ello, se emplean estrategias estándar de extensión multiclase, como los esquemas uno-contra-resto (one-vs-rest) o uno-contra-uno (one-vs-one), que combinan varias

SVM binarias para obtener una decisión final entre múltiples clases. La elección de la estrategia concreta se especifica también en la metodología del Capítulo 3.

Las SVM presentan varias características que justifican su elección como clasificador unificado en este TFG:

- **Adecuación a vectores de alta dimensión:** los descriptores considerados (histogramas LBP, características de Haralick, energías de Gabor, embeddings de CNN) suelen generar vectores de características de tamaño medio o elevado; la SVM está especialmente diseñada para manejar estos espacios, mitigando el riesgo de sobreajuste mediante el control del margen y la regularización.
- **Buen comportamiento con conjuntos de datos moderados:** a diferencia de muchos modelos profundos, la SVM puede alcanzar un rendimiento competitivo con cantidades de datos de entrenamiento relativamente limitadas, escenario habitual en problemas de visión industrial.
- **Flexibilidad mediante kernels:** la posibilidad de utilizar distintos kernels permite adaptar la complejidad del modelo al tipo de descriptor utilizado sin necesidad de modificar la estructura del clasificador.
- **Amplia adopción en la literatura de texturas:** numerosos trabajos utilizan SVM como clasificador de referencia sobre descriptores clásicos y, más recientemente, sobre embeddings derivados de CNN, lo que facilita la comparación y contextualización de los resultados obtenidos.

En consecuencia, en este TFG la SVM se adopta como “juez único” en la comparación entre descriptores clásicos y representaciones generadas por CNN. Cada método de representación produce un vector de características de dimensión fija que se normaliza y suministra a una SVM configurada de acuerdo con el protocolo experimental descrito en el Capítulo 3. Este enfoque permite aislar, en la medida de lo posible, el efecto de la representación sobre el rendimiento, manteniendo constante el modelo de decisión.

2.3.2. Otros clasificadores (k-NN, Random Forest, etc.) como contexto

Además de las SVM, la literatura recoge numerosas alternativas de clasificación supervisada que han sido empleadas en combinación con descriptores de textura. Aunque en el presente TFG no constituyen el foco principal del análisis comparativo,

su mención resulta pertinente para situar el trabajo en el contexto más amplio de la clasificación de patrones.

Entre los clasificadores tradicionales más habituales cabe destacar:

- **k-Nearest Neighbors (k-NN):** se trata de un método no paramétrico que asigna a cada muestra de prueba la clase mayoritaria entre sus k vecinos más cercanos en el espacio de características, de acuerdo con una métrica (típicamente la distancia euclídea). Su principal ventaja es la simplicidad conceptual y de implementación; sin embargo, su coste computacional en clasificación crece con el número de muestras de entrenamiento y su rendimiento puede verse afectado por la dimensión del espacio de características.

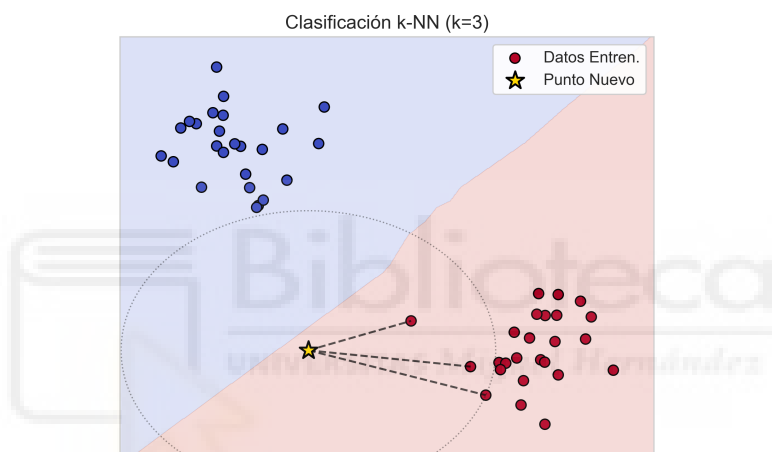


Figura 2.9: Representación esquemática de la regla de decisión del algoritmo k-Nearest Neighbors (k-NN) configurado con $k = 3$. El punto a clasificar (estrella amarilla) se evalúa calculando la distancia respecto a las muestras de entrenamiento. La decisión final se toma por votación mayoritaria entre los k vecinos más cercanos (conectados por líneas discontinuas); en este ejemplo, dado que los tres vecinos más próximos pertenecen a la Clase B (rojos), el nuevo punto es asignado a dicha clase.

- **Modelos lineales discriminativos:** entre ellos, la regresión logística y los clasificadores lineales basados en funciones de pérdida convexas (por ejemplo, perceptrones con regularización). Estos modelos aprenden hiperplanos de separación en el espacio de características y ofrecen interpretabilidad y eficiencia computacional, pero pueden resultar demasiado restrictivos cuando las relaciones entre características y clases son marcadamente no lineales.

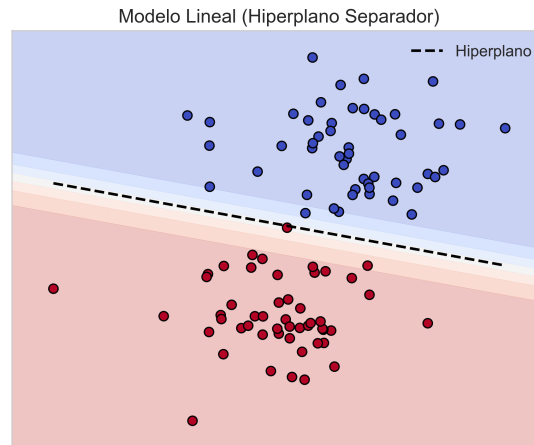


Figura 2.10: Visualización de un clasificador lineal (Regresión Logística). La frontera de decisión es una recta (hiperplano en dimensiones superiores) que divide el espacio en dos regiones. Este modelo es eficiente pero incapaz de separar datos con estructuras curvas o complejas.

- **Métodos basados en árboles de decisión y ensembles:** los Random Forest combinan múltiples árboles de decisión entrenados sobre subconjuntos aleatorios de datos y características, y agregan sus predicciones para mejorar la robustez y reducir el sobreajuste. Este tipo de métodos puede manejar relaciones no lineales y mezclas de características heterogéneas, y proporciona medidas de importancia de variables, aunque su comportamiento con vectores de muy alta dimensión y altamente correlacionados depende de la configuración concreta del ensemble.

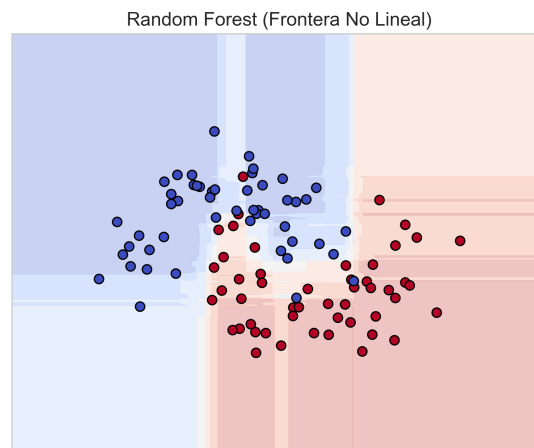


Figura 2.11: Frontera de decisión generada por un Random Forest sobre datos no linealmente separables. A diferencia de los modelos lineales, el ensemble de árboles puede modelar contornos complejos y regiones de decisión discontinuas, adaptándose mejor a la estructura intrínseca de los datos.

- **Modelos probabilísticos sencillos:** como Naïve Bayes, que asumen inde-

pendencia condicional entre características. A pesar de la simplicidad y de que sus hipótesis rara vez se cumplen de forma estricta en problemas reales, pueden ofrecer un rendimiento razonable con bajo coste en escenarios donde la estructura de los datos es relativamente simple.

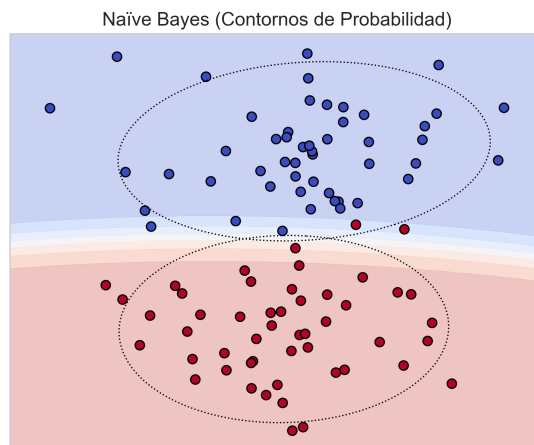


Figura 2.12: Representación de un clasificador Naïve Bayes Gaussiano. El modelo estima las distribuciones de probabilidad subyacentes de cada clase (representadas por las elipses punteadas y el gradiente de color). La decisión de clasificación se basa en la probabilidad posterior de pertenencia a cada clase.

Por estas razones, en este TFG se ha optado por utilizar la SVM como clasificador de referencia para todas las comparativas cuantitativas entre descriptores clásicos y representaciones derivadas de CNN. El resto de los clasificadores tradicionales se consideran únicamente como contexto metodológico y como posibles alternativas a explorar en trabajos futuros, pero no forman parte del “torneo” experimental principal definido en la metodología.

2.4. Redes neuronales convolucionales (CNN): fundamentos

Las redes neuronales convolucionales (Convolutional Neural Networks, CNN) constituyen hoy en día uno de los pilares del aprendizaje profundo en visión por computador. Se trata de modelos neurales diseñados específicamente para procesar datos con estructura espacial, como imágenes, aprovechando propiedades como la localidad y la invariancia traslacional mediante el uso de filtros convolucionales y operaciones de subsampling.

Desde un punto de vista funcional, una CNN recibe como entrada una imagen (o un lote de imágenes) y aplica una sucesión de capas que transforman progresivamente la representación de los datos, pasando de píxeles brutos a descriptores de alto

nivel útiles para tareas de clasificación, segmentación o detección. En los primeros niveles suelen aparecer filtros sensibles a bordes, orientaciones o texturas simples, mientras que en niveles más profundos se capturan patrones cada vez más complejos y abstractos.

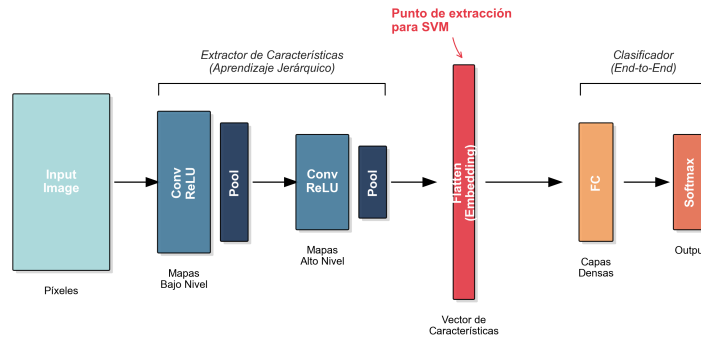


Figura 2.13: Diagrama esquemático de una Red Neuronal Convolutiva (CNN). El bloque azul representa la etapa de extracción de características, donde los filtros convolucionales transforman la imagen en mapas de activación. El bloque rojo (Flatten/Embedding) representa el vector de características latente: en un entrenamiento *end-to-end*, este vector alimenta a las capas densas (naranja) para clasificar; en la metodología híbrida de este TFG, este vector se extrae para entrenar una SVM externa.

En este TFG, las CNN cumplen un doble papel:

- por un lado, se utilizan como modelos end-to-end, donde la red aprende conjuntamente la representación y el clasificador final;
- por otro, se emplean como extractores de características, en los que se toman las activaciones de capas intermedias o finales (embeddings) para suministrarlas a un clasificador externo, en particular a la SVM adoptada como “juez unificado” en la comparación con descriptores clásicos.

2.4.1. Arquitectura básica de una CNN

Una CNN típica está formada por la composición de distintos tipos de capas:

- **Capas convolucionales:** aplican conjuntos de filtros (kernels) que se deslizan sobre la imagen. Dada una entrada I y un filtro K , la operación de convolución discreta 2D se define como:

$$S(i, j) = (I * K)(i, j) = \sum_u \sum_v I(i + u, j + v) K(u, v) \quad (2.5)$$

Esta operación produce un mapa de características que responde a ciertos patrones locales (por ejemplo, bordes horizontales, texturas estriadas, esquinas). Los pesos de estos filtros se aprenden durante el entrenamiento, de modo que la red descubre automáticamente qué patrones son relevantes para la tarea.

- **Funciones de activación no lineales:** tras la operación de convolución se aplica normalmente una función no lineal punto a punto, como ReLU (Rectified Linear Unit), que introduce no linealidad en el modelo y permite representar funciones complejas.
- **Capas de pooling o subsampling:** reducen la resolución espacial de los mapas de activación combinando localmente grupos de valores (por ejemplo, mediante max pooling o average pooling). Este subsampling aporta cierta invariancia a traslaciones pequeñas y reduce el coste computacional, a costa de perder parte de la información de detalle.
- **Capas totalmente conectadas (fully connected):** en las arquitecturas clásicas de clasificación, tras varias capas convolucionales y de pooling se incluye uno o varios niveles totalmente conectados que operan sobre una versión vectorizada de las características extraídas. Estas capas actúan como clasificadores de alto nivel, mapeando la representación aprendida a las puntuaciones de cada clase.

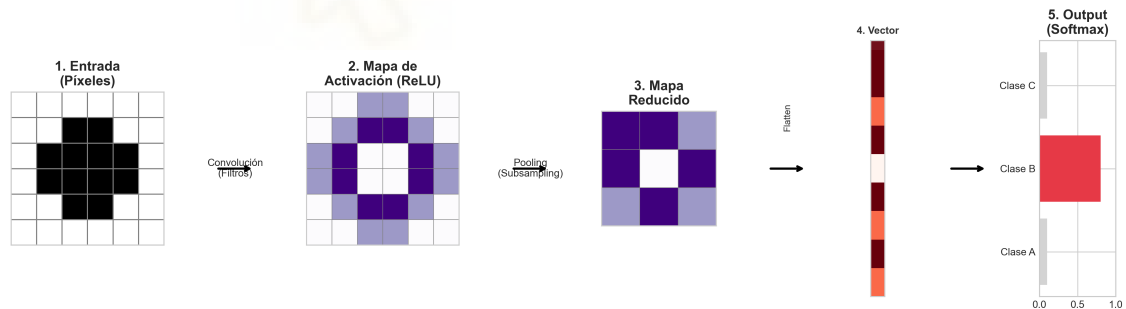


Figura 2.14: Flujo de transformación de datos en una arquitectura CNN básica. (1) La imagen de entrada es procesada por **capas convolucionales** que aplican filtros para detectar patrones locales. (2) La función de activación **ReLU** introduce no linealidad. (3) Las capas de **pooling** reducen la dimensionalidad espacial conservando la información más relevante. (4) Finalmente, los mapas de características se aplanan (*flatten*) para alimentar las **capas totalmente conectadas** que generan la predicción final (5).

En muchas arquitecturas modernas se han introducido variaciones sobre este esquema básico, tales como capas de normalización (Batch Normalization), bloques residuales, conexiones de salto o mecanismos de atención. Sin embargo, la idea central

se mantiene: la red aplica filtros locales compartidos espacialmente que explotan la estructura bidimensional de las imágenes y construyen representaciones jerárquicas a partir de los datos.

En el contexto de texturas, las primeras capas convolucionales pueden interpretarse como análogas a bancos de filtros de Gabor u otros detectores de patrones locales, mientras que las capas más profundas combinan estas respuestas para capturar estructuras repetitivas o patrones de mayor escala característicos de cada categoría de textura.

2.4.2. Entrenamiento y aprendizaje de representaciones jerárquicas

El entrenamiento de una CNN se plantea como un problema de aprendizaje supervisado: se dispone de un conjunto de imágenes etiquetadas, y el objetivo es ajustar los pesos de la red para minimizar una función de pérdida que cuantifica el error de clasificación. Este proceso se lleva a cabo mediante variantes del algoritmo de retropropagación del error (backpropagation) combinado con métodos de optimización como descenso de gradiente estocástico (Stochastic Gradient Descent, SGD) y sus extensiones (por ejemplo, Adam, RMSprop).

Durante el entrenamiento, la red aprende automáticamente filtros que responden a patrones relevantes para la tarea final. Este aprendizaje jerárquico presenta varias características de interés:

- En las primeras capas, los filtros suelen detectar patrones genéricos (bordes, esquinas, cambios de intensidad, texturas simples) que aparecen de forma recurrente en muchas imágenes, independientemente de la clase.
- En capas intermedias, se combinan estas primitivas para formar descriptores más específicos (motivos, estructuras repetitivas, combinaciones de texturas).
- En las últimas capas, la representación se especializa en separar las clases del problema concreto, produciendo embeddings que facilitan la tarea de clasificación.

Este comportamiento ha motivado el uso de CNN preentrenadas en grandes bases de datos generales (como ImageNet) y su posterior adaptación a tareas específicas mediante aprendizaje transferido (transfer learning), bien ajustando solo las últimas capas, bien reutilizando la red como extractor de características y entrenando un clasificador externo sobre los embeddings generados.

En el caso de la clasificación de texturas, esta idea es especialmente relevante: aunque las texturas del DTD difieren de los objetos de ImageNet, las primeras capas de una CNN SOTA siguen proporcionando filtros que capturan bordes, orientaciones y patrones locales útiles, mientras que las capas intermedias y finales pueden adaptarse al dominio de texturas mediante fine-tuning o reentrenamiento parcial. En este TFG se exploran precisamente estas posibilidades, comparando:

- CNN entrenadas end-to-end sobre DTD;
- embeddings de CNN preentrenadas utilizados como descriptores fijos alimentados a una SVM;
- y descriptores clásicos (LBP, Haralick, Gabor) tratados de forma análoga.

2.4.3. Ventajas y limitaciones en el contexto de texturas y automatización

Las CNN han demostrado un rendimiento sobresaliente en un amplio abanico de tareas de visión por computador, incluyendo la clasificación de texturas, y presentan varias ventajas clave frente a los descriptores clásicos:

- **Aprendizaje automático de características:** no requieren diseñar manualmente filtros o estadísticas específicas para cada problema; la red aprende directamente, a partir de los datos, qué patrones son relevantes para la clasificación.
- **Capacidad de modelar relaciones no lineales complejas:** la composición de capas convolucionales y no lineales permite representar funciones altamente no lineales, adaptándose a variaciones de iluminación, deformaciones y cambios de apariencia más allá de lo que capturan muchos descriptores manuales.
- **Representaciones jerárquicas reutilizables:** las capas inferiores y medias de una CNN entrenada en un dominio amplio pueden reutilizarse en otras tareas mediante aprendizaje transferido, reduciendo los requisitos de datos y tiempo de entrenamiento.

Sin embargo, desde la perspectiva de la ingeniería de automatización, las CNN presentan también limitaciones y desafíos relevantes:

- **Coste computacional y de memoria:** las arquitecturas profundas, especialmente las diseñadas para grandes benchmarks, pueden requerir recursos de cómputo significativos (GPU u otros aceleradores), lo que dificulta su despliegue directo en dispositivos embebidos con capacidad limitada.
- **Necesidad de datos etiquetados:** el entrenamiento eficaz de modelos profundos suele requerir grandes cantidades de datos anotados; aunque el aprendizaje transferido mitiga parcialmente este problema, la disponibilidad de datos representativos sigue siendo un factor crítico.
- **Complejidad y explicabilidad:** las CNN se perciben a menudo como “cajas negras” difíciles de interpretar. Aunque existen técnicas de explicabilidad (como Grad-CAM y otros métodos de atribución) que permiten visualizar regiones relevantes de la imagen, la interpretación de estas visualizaciones requiere cuidado y no ofrece, por sí sola, garantías formales sobre el comportamiento del modelo.
- **Sensibilidad a distribuciones cambiantes:** como cualquier modelo estadístico, las CNN pueden ver degradado su rendimiento si las condiciones de captura en producción difieren significativamente de las del conjunto de entrenamiento (cambios en iluminación, tipo de cámara, ruido, etc.).

Estos aspectos hacen que, en aplicaciones de automatización, la decisión entre utilizar CNN o descriptores clásicos no pueda basarse únicamente en la precisión de clasificación. Es necesario considerar la eficiencia computacional y la explicabilidad, tal y como se ha planteado en la Introducción. En este TFG, las CNN se integran en un marco comparativo que:

- evalúa su rendimiento sobre DTD en condiciones estándar y bajo perturbaciones controladas,
- mide sus costes de inferencia en un entorno representativo,
- y analiza su comportamiento mediante técnicas de explicabilidad visual,

todo ello en comparación con descriptores clásicos evaluados bajo el mismo protocolo, utilizando principalmente una SVM como clasificador unificado. De este modo, las capacidades de las CNN se sitúan en un contexto realista de ingeniería de automatización basada en texturas, complementando el análisis puramente algorítmico con criterios de viabilidad práctica.

2.5. Arquitecturas convolucionales de última generación (SOTA)

Desde el éxito de las primeras redes neuronales convolucionales profundas en competiciones como ImageNet, se han propuesto numerosas arquitecturas que han ido ampliando el estado del arte en clasificación de imágenes. Estas arquitecturas se caracterizan por una combinación de mayor profundidad, bloques estructurales específicos (por ejemplo, conexiones residuales, bloques Inception, densas conexiones), técnicas de normalización y estrategias de regularización que permiten entrenar modelos muy complejos de forma estable.

En términos generales, la evolución de las arquitecturas SOTA ha seguido dos líneas complementarias:

- Aumento de profundidad y capacidad, buscando redes más expresivas capaces de capturar patrones visuales complejos.
- Mejoras de eficiencia y escalado, intentando mantener o mejorar el rendimiento con un coste computacional contenido, especialmente relevante para despliegue en dispositivos con recursos limitados.

En esta sección se revisan de forma sintética las arquitecturas más influyentes en la literatura de visión por computador, con especial atención a dos familias que se han convertido en referencia en numerosos trabajos: ResNet y EfficientNet.

2.5.1. Evolución de las arquitecturas CNN SOTA

El punto de inflexión en la adopción masiva de CNN en visión por computador fue la arquitectura AlexNet, propuesta por Krizhevsky et al., que obtuvo una mejora significativa respecto a los métodos tradicionales en la competición ImageNet 2012. AlexNet introdujo varias ideas clave para la época, como el uso extensivo de la función de activación ReLU, el entrenamiento distribuido en GPU y técnicas de regularización como dropout.

Posteriormente, la arquitectura VGG de Simonyan y Zisserman profundizó en la idea de redes muy profundas compuestas por bloques sencillos de convoluciones (3×3) y pooling, demostrando que aumentar la profundidad, incluso con bloques de estructura homogénea, podía mejorar de forma sustancial la precisión en clasificación de

imágenes. Sin embargo, VGG supone un coste computacional elevado, especialmente en términos de parámetros.

En paralelo, la serie de arquitecturas GoogLeNet/Inception introdujo los bloques Inception, que combinan en paralelo convoluciones de distintos tamaños y rutas de procesamiento dentro de un mismo bloque, permitiendo capturar patrones a diferentes escalas con un coste de parámetros más contenido. Esta línea de trabajo puso de relieve la importancia del diseño estructural de los bloques convolucionales para equilibrar precisión y eficiencia.

Sobre estas bases se desarrollaron arquitecturas más avanzadas que incorporan:

- Conexiones de salto (skip connections) y bloques residuales, como en las ResNet, que facilitan el entrenamiento de redes muy profundas.
- Conexiones densas entre capas, como en DenseNet, que favorecen la reutilización de características y la propagación eficiente del gradiente.
- Estrategias de escalado compuesto y búsqueda de arquitecturas eficientes, como en EfficientNet, que equilibran profundidad, ancho y resolución de entrada de forma sistemática.

Estas familias de arquitecturas han sido ampliamente utilizadas como modelos base en tareas de clasificación de imágenes, incluyendo la clasificación de texturas, tanto en configuración end-to-end como como extractores de características para clasificadores externos.

2.5.2. Redes residuales (ResNet)

Las Redes Residuales (Residual Networks, ResNet) fueron propuestas por He et al. como respuesta al problema de degradación del rendimiento observado al aumentar la profundidad de las redes. A partir de cierta profundidad, redes puramente apiladas (como VGG) resultaban difíciles de entrenar: no solo no mejoraban, sino que podían empeorar la precisión en el conjunto de validación, incluso con técnicas de normalización y regularización estándar.

La idea central de ResNet es introducir bloques residuales que aprendan no una función directa $H(x)$, sino una corrección residual $F(x) = H(x) - x$. Para ello, se añade una conexión de salto (shortcut) que suma la entrada del bloque con la salida

de una serie de capas convolucionales:

$$y = F(x) + x$$

donde $F(x)$ representa la transformación no lineal aprendida por las capas convolucionales del bloque. Esta estructura presenta varias ventajas:

- facilita la propagación del gradiente hacia capas profundas, reduciendo problemas de desvanecimiento;
- permite que el bloque aprenda, en el peor caso, la identidad, evitando que la mayor profundidad degrade el rendimiento;
- favorece la reutilización de características a lo largo de la red.

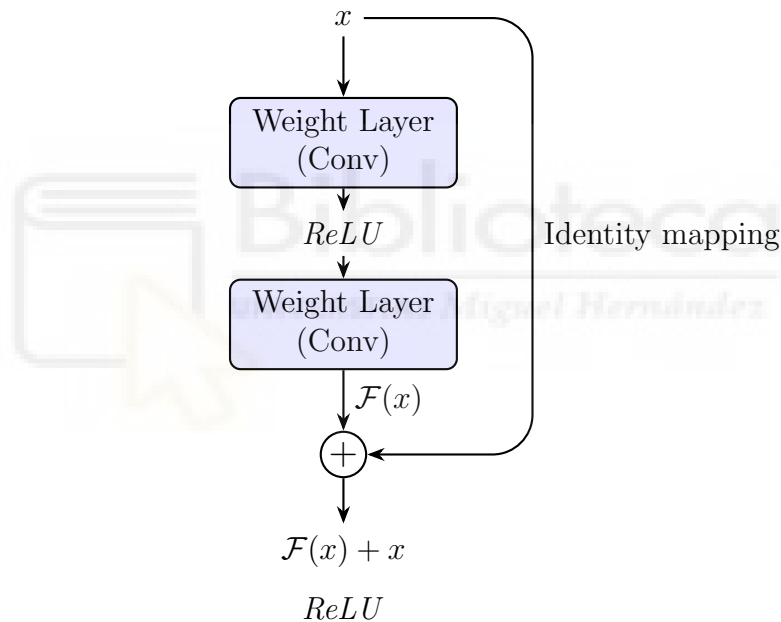


Figura 2.15: Esquema de un bloque residual básico. La conexión de salto (*shortcut*) realiza una identidad que se suma a la salida de las capas de convolución $\mathcal{F}(x)$. Esta arquitectura permite mitigar el problema del desvanecimiento del gradiente.

Existen diferentes variantes de ResNet, caracterizadas por el número de capas (ResNet-18, ResNet-34, ResNet-50, ResNet-101, etc.) y por el tipo de bloque residual empleado (bloques básicos frente a bloques bottleneck). En particular, ResNet-50 se ha convertido en una de las arquitecturas de referencia más utilizadas en la literatura y en aplicaciones prácticas, ya que ofrece un compromiso razonable entre profundidad, capacidad de representación y coste computacional.

Las propiedades que hacen a ResNet especialmente relevante en el contexto de este TFG son:

- su capacidad para aprender representaciones ricas en profundidad, lo que resulta útil para capturar patrones de textura complejos que combinan información a múltiples escalas;
- su amplio uso como modelo base preentrenado en grandes conjuntos de datos (como ImageNet), lo que facilita el uso de estrategias de aprendizaje transferido y de extracción de embeddings para tareas específicas, como la clasificación de texturas.

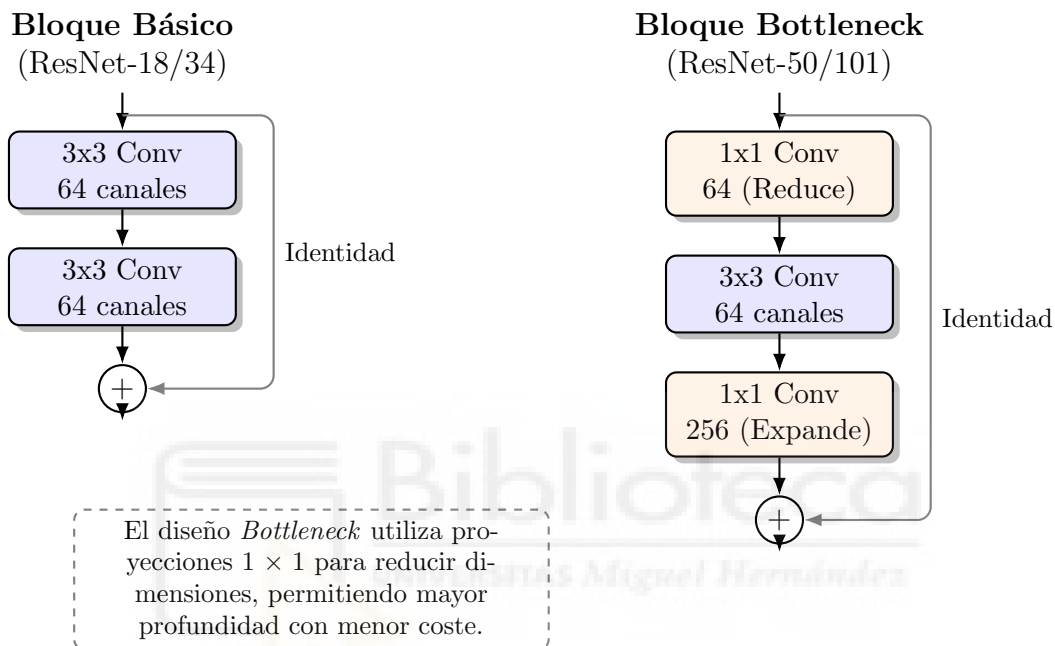


Figura 2.16: Comparación arquitectónica entre el Bloque Residual Básico (izquierda) y el Bloque *Bottleneck* (derecha) utilizado en ResNet-50. Nótese la estructura de "sándwich" (1x1, 3x3, 1x1) en la derecha que optimiza el cómputo.

Sin entrar aún en la justificación de su uso concreto en este trabajo (que se detalla en el Capítulo 3), ResNet representa una arquitectura SOTA representativa de las redes residuales profundas actualmente empleadas en visión por computador.

2.6. Aprendizaje transferido y fine-tuning en visión por computador

El aprendizaje transferido (transfer learning) se ha consolidado como una estrategia fundamental en visión por computador para aprovechar el conocimiento adquirido por modelos entrenados en grandes conjuntos de datos genéricos (como ImageNet) y reutilizarlo en tareas específicas con disponibilidad limitada de datos. La idea central consiste en partir de una red previamente entrenada en un problema fuente y

adaptarla a un problema destino relacionado, reduciendo así el esfuerzo de entrenamiento y mejorando el rendimiento cuando el conjunto de datos destino no es lo suficientemente grande como para entrenar una red profunda desde cero.

En el contexto de las CNN, el aprendizaje transferido se basa en la observación de que:

- las primeras capas aprenden filtros relativamente genéricos (bordes, orientaciones, texturas simples);
- las capas intermedias codifican motivos y estructuras que pueden seguir siendo útiles en distintos dominios;
- las capas finales se especializan en las clases concretas del problema fuente.

Esta estructura jerárquica hace posible reutilizar gran parte de la red como extractor de características y limitar la adaptación a las capas superiores, lo que resulta especialmente ventajoso cuando el conjunto de datos destino, como un dataset de texturas en aplicaciones de automatización, es de tamaño moderado.

En los últimos años, el aprendizaje transferido ha demostrado su eficacia en numerosas tareas de visión por computador, incluyendo clasificación de objetos, detección, segmentación y, de forma específica, clasificación de texturas, donde modelos preentrenados en ImageNet se han usado como base para obtener descriptores robustos a partir de arquitecturas SOTA como ResNet o EfficientNet.

2.6.1. Concepto general de aprendizaje transferido en CNN

Formalmente, en aprendizaje transferido se distingue entre un dominio fuente D_S y una tarea fuente T_S , frente a un dominio destino D_T y una tarea destino T_T . En el caso típico de visión:

- D_S corresponde a un gran conjunto de imágenes naturales (por ejemplo, ImageNet) con una distribución amplia y variada;
- T_S es una tarea de clasificación general con un gran número de clases (miles de categorías de objetos);
- D_T representa el conjunto de imágenes específico de la aplicación (por ejemplo, texturas del DTD);

- T_T es la tarea de clasificación de interés (por ejemplo, asignar cada textura a una de las 47 clases de DTD).

En este esquema, una CNN SOTA se entrena primero sobre D_S , aprendiendo filtros y representaciones generales. A continuación, este modelo se adapta a D_T mediante alguna de las estrategias de transferencia descritas en la siguiente subsección.

La eficacia de la transferencia depende de diversos factores, entre ellos:

- el grado de similitud entre el dominio fuente y el dominio destino (por ejemplo, texturas “en el mundo real” frente a objetos naturales);
- el tamaño y la diversidad del conjunto de datos destino;
- la profundidad hasta la que se adaptan (o no) los pesos de la red;
- y el tipo de capas cuyos embeddings se reutilizan (capas intermedias frente a capas muy finales).

Numerosos trabajos han mostrado que, incluso cuando la tarea destino difiere considerablemente de la tarea fuente, las primeras capas convolucionales de redes entrenadas sobre conjuntos grandes siguen siendo útiles como extractores de características genéricas, mientras que las capas superiores requieren una adaptación más cuidadosa.

2.6.2. Estrategias habituales: extracción de características vs fine-tuning

En la práctica, el aprendizaje transferido con CNN se implementa principalmente mediante dos estrategias:

Extracción de características fija (feature extraction) En este enfoque, la CNN preentrenada se utiliza como un bloque fijo que transforma las imágenes de entrada en vectores de características (embeddings). Concretamente:

- se elimina o ignora la capa de clasificación original de la red;
- se selecciona una capa intermedia o penúltima capa como fuente de las características;
- se congela el resto de la red (no se actualizan sus pesos);

- y se entrena un clasificador externo (por ejemplo, una SVM o una pequeña red totalmente conectada) sobre los embeddings generados para el conjunto destino.

Esta estrategia presenta varias ventajas en contextos con recursos limitados o pocos datos etiquetados:

- reduce el número de parámetros que deben ajustarse, disminuyendo el riesgo de sobreajuste;
- simplifica el entrenamiento (solo se optimizan las últimas capas o el clasificador externo);
- facilita la comparación entre distintos métodos de representación, ya que el clasificador puede mantenerse fijo.

En el contexto del presente TFG, esta estrategia se relaciona directamente con la idea de utilizar arquitecturas SOTA como extractores de descriptores de textura, cuyas salidas alimentan a una SVM común, en paralelo a los descriptores clásicos (Sección 2.2), aunque el uso concreto se detallará más adelante en la Sección 2.7 y en la Metodología del Capítulo 3.

Ajuste fino (fine-tuning) En el enfoque de fine-tuning, la CNN preentrenada se toma como punto de partida, pero se permite que algunas de sus capas (normalmente las superiores) se adapten durante el entrenamiento en el dominio destino. En muchos casos:

- se inicializa la red con los pesos preentrenados;
- se reemplaza la última capa de clasificación por una nueva capa ajustada al número de clases de la tarea destino;
- se congelan inicialmente las capas más bajas y se entrena solo el nuevo “cabezal” de clasificación;
- y, opcionalmente, se descongelan progresivamente capas intermedias o superiores para refinar la representación de la red de acuerdo con los datos destino.

Esta estrategia permite que el modelo adapte sus representaciones internas al dominio específico (en este caso, texturas del DTD), potencialmente mejorando el rendimiento respecto a la extracción de características fija, a costa de:

- un mayor coste computacional de entrenamiento;
- y un mayor riesgo de sobreajuste si el conjunto de datos destino es pequeño o poco diverso.

En la práctica, la elección entre feature extraction y fine-tuning depende de factores como el tamaño del conjunto de datos destino, la similitud entre dominios y las restricciones de tiempo de entrenamiento y recursos de cómputo. En trabajos recientes sobre clasificación de texturas, ambas estrategias se han utilizado para evaluar la capacidad de redes preentrenadas (por ejemplo, VGG, ResNet, DenseNet, EfficientNet) de proporcionar descriptores discriminativos para texturas “en el mundo real”, ya sea entrenando un nuevo clasificador sobre embeddings fijos o reajustando parcialmente la red.

2.7. CNN como extractores de características: embeddings y capas intermedias

Como se ha expuesto en las secciones anteriores, las redes neuronales convolucionales (CNN) aprenden representaciones jerárquicas de las imágenes, desde patrones locales simples hasta descriptores de alto nivel especializados en la tarea de clasificación. Esta estructura hace posible interpretar una CNN no solo como un modelo de clasificación end-to-end, sino también como una función de extracción de características que transforma una imagen de entrada en un vector de alto nivel (un embedding) adecuado para ser utilizado por clasificadores externos.

En esta sección se revisa el papel de las CNN como extractores de descriptores de textura, enfatizando la idea de que las activaciones de capas intermedias o finales pueden tratarse como vectores de características análogos, en espíritu, a los generados por descriptores clásicos como LBP, Haralick o Gabor (Sección 2.2), pero aprendidos de manera automática a partir de datos.

2.7.1. Embeddings derivados de capas intermedias y finales

Desde un punto de vista abstracto, una CNN puede verse como una función paramétrica

$$\phi : \mathbb{R}^{H \times W \times C} \rightarrow \mathbb{R}^d$$

que mapea una imagen de tamaño $H \times W$ y C canales a un vector de características $\phi(x) \in \mathbb{R}^d$. En las arquitecturas de clasificación estándar, este vector suele corres-

ponder a:

- la salida de la penúltima capa totalmente conectada, previa a la capa de clasificación softmax;
- o el resultado de aplicar una operación de pooling global (por ejemplo, global average pooling) sobre los mapas de activación de la última capa convolucional, obteniendo un vector de dimensión igual al número de canales.

En el marco del aprendizaje transferido (Sección 2.6), la red se entrena inicialmente en una tarea fuente (por ejemplo, clasificación de objetos en ImageNet), y después se reutiliza la función $\phi(\cdot)$ para generar embeddings en una tarea destino (por ejemplo, clasificación de texturas en DTD). Estos embeddings condensan información de alto nivel sobre la imagen, integrando respuestas de múltiples filtros a distintas escalas y orientaciones, y pueden emplearse como descriptores de textura aprendidos.

2.8. Explicabilidad en visión por computador

La explicabilidad en el ámbito de la visión por computador (Explainable AI o XAI) se refiere al conjunto de técnicas y métodos que permiten comprender y visualizar las decisiones tomadas por modelos complejos, como las redes neuronales convolucionales (CNN). Dado que estos modelos funcionan a menudo como “cajas negras”, resulta fundamental desarrollar herramientas que identifiquen qué regiones de una imagen o qué patrones visuales han sido determinantes para una predicción concreta. Esto es especialmente crítico en aplicaciones industriales y de automatización, donde la validación y la confianza en el sistema son requisitos indispensables.

2.8.1. Grad-CAM y variantes

Una de las técnicas más destacadas y ampliamente utilizadas para la visualización de decisiones en CNN es Grad-CAM (Gradient-weighted Class Activation Mapping). Propuesta por Selvaraju et al., esta técnica utiliza la información de los gradientes que fluyen hacia la última capa convolucional de la red para entender la importancia de cada neurona en la decisión final.

La idea central de Grad-CAM es calcular el gradiente de la puntuación de la clase predicha con respecto a los mapas de características de la última capa convolucional. Estos gradientes se promedian globalmente para obtener unos pesos de importancia,

que indican cuánto contribuye cada mapa de características a la clase de interés. Posteriormente, se realiza una suma ponderada de los mapas de activación utilizando estos pesos y se aplica una función ReLU para eliminar las contribuciones negativas (es decir, aquellas que reducen la confianza en la clase). El resultado es un mapa de calor (heatmap) que resalta las regiones de la imagen original que más han influido en la clasificación.

Grad-CAM presenta la ventaja de ser agnóstica a la arquitectura de la red, siempre que esta sea diferenciable, y no requiere reentrenamiento. Además, se han desarrollado variantes como Grad-CAM++, que mejoran la localización de objetos múltiples o dispersos en la imagen, proporcionando mapas de activación más precisos en escenarios complejos.

2.8.2. Otras técnicas de XAI visual

Además de los métodos basados en gradientes como Grad-CAM, existen otras aproximaciones para la explicabilidad en visión por computador que ofrecen perspectivas complementarias:

- **Perturbaciones y Oclusión (Occlusion Sensitivity):** Este método consiste en ocultar sistemáticamente diferentes partes de la imagen mediante un parche gris o borroso y observar cómo varía la probabilidad de la clase correcta. Si al ocultar una región la confianza del modelo cae drásticamente, se infiere que esa zona era crítica para la predicción. Es una técnica intuitiva pero computacionalmente costosa.
- **LIME (Local Interpretable Model-agnostic Explanations):** LIME aproxima el comportamiento del modelo complejo mediante un modelo localmente interpretable (como una regresión lineal) alrededor de la predicción de interés. Genera explicaciones perturbando la entrada (superpíxeles de la imagen) y analizando cómo afectan estos cambios a la salida.
- **SHAP (SHapley Additive exPlanations):** Basado en la teoría de juegos, SHAP asigna a cada píxel o superpíxel un valor de importancia (valor de Shapley) que representa su contribución marginal a la predicción final. Ofrece un marco teórico sólido y consistente para la atribución de importancia.

Estas técnicas permiten a los ingenieros y expertos en automatización verificar si el modelo está aprendiendo los patrones correctos (por ejemplo, la textura de un

material) o si se está basando en correlaciones espurias o sesgos del conjunto de datos, lo cual es vital para el despliegue seguro de sistemas de visión artificial.

2.9. Dataset de trabajo: Describable Textures Dataset (DTD)

El Describable Textures Dataset (DTD) es el conjunto de datos de referencia utilizado en este Trabajo de Fin de Grado para evaluar y comparar tanto los descriptores clásicos de textura como las arquitecturas de redes neuronales convolucionales. Fue propuesto por Cimpoi et al. como un dataset de texturas “en el mundo real” descritas mediante atributos semánticos, con el objetivo de estudiar la relación entre descriptores de textura y conceptos lingüísticos cotidianos.



Figura 2.17: Muestra aleatoria de categorías del Describable Textures Dataset (DTD). Se muestran 12 ejemplos extraídos del conjunto de datos local, abarcando clases semánticas como *banded* (bandedado), *honeycombed* (apanalado) o *fibrous* (fibroso). Nótese la alta variabilidad intra-clase en términos de iluminación, perspectiva y escala, lo que convierte a este dataset en un desafío para los algoritmos de clasificación.

DTD contiene 5 640 imágenes de textura a color, recolectadas a partir de fotografías

de escenas naturales y de objetos reales, que se organizan en 47 categorías etiquetadas con adjetivos descriptivos tales como *banded*, *blotchy*, *cracked*, *dotted*, *striped*, *zigzagged*, entre otros. Cada categoría agrupa aproximadamente 120 imágenes, lo que proporciona un número moderado de ejemplos por clase para tareas de clasificación supervisada.

A diferencia de los datasets clásicos de texturas adquiridas en condiciones muy controladas (por ejemplo, sobre superficies planas iluminadas de forma homogénea), DTD se caracteriza por:

- incluir texturas en contextos “in-the-wild”, con variaciones significativas de iluminación, escala, perspectiva y fondo,
- presentar texturas parciales o mezcladas con otras estructuras de la escena,
- y reflejar condiciones de captura realistas (ruido, desenfoque moderado, oclusiones parciales).

Estas características hacen de DTD un banco de pruebas exigente y representativo para estudiar la capacidad de generalización de descriptores de textura y modelos de aprendizaje profundo en escenarios cercanos a aplicaciones reales.

3. Metodología y diseño experimental

En este capítulo se describe la metodología seguida para comparar, de forma sistemática y bajo un marco experimental común, distintas familias de métodos para la clasificación de texturas en el Describable Textures Dataset (DTD). El diseño experimental busca aislar, en la medida de lo posible, el papel de la representación de características frente al del clasificador, y responder a las preguntas de investigación planteadas en la Introducción en términos de rendimiento, eficiencia y explicabilidad.

3.1. Justificación metodológica del enfoque basado en un clasificador unificado (SVM)

La literatura sobre clasificación de texturas presenta una gran variedad de combinaciones entre descriptores clásicos (como LBP, características de Haralick o filtros de Gabor), arquitecturas profundas (VGG, ResNet, EfficientNet, etc.) y clasificadores (SVM, k-NN, redes totalmente conectadas, Random Forest, entre otros). En muchos trabajos las comparaciones se realizan enfrentando pipelines completos del tipo “descriptor X + clasificador Y” frente a “arquitectura Z end-to-end”, lo que implica comparar simultáneamente la calidad de la representación y la potencia del modelo de decisión.

Este enfoque presenta una limitación importante para los objetivos de este TFG: cuando se observa que un método supera a otro, no resulta sencillo discernir si la mejora se debe principalmente al descriptor, al clasificador o a la interacción concreta entre ambos. En consecuencia, es difícil extraer conclusiones claras sobre qué representación de textura resulta más adecuada para un determinado contexto de automatización.

Para mitigar este problema se adopta aquí un enfoque metodológico basado en un clasificador unificado. En concreto, se selecciona una Máquina de Vectores de Soporte (SVM) con kernel RBF como clasificador de referencia y se utilizan configuraciones equivalentes de SVM para todos los vectores de características considerados. Desde este punto de vista, cada método evaluado se interpreta como un módulo de extracción de características que transforma una imagen en un vector de dimensión fija, que posteriormente es clasificado por la misma SVM.

3.2. Bloques de descriptores y esquema comparativo

A partir de las consideraciones anteriores, la comparación experimental se organiza en torno a varios bloques de métodos, definidos según el tipo de representación que generan. La idea es agrupar en un mismo bloque aquellos métodos que comparten una filosofía similar de descripción de la textura, aun cuando sus detalles concretos difieran.

3.2.1. Bloque 1 – Descriptores clásicos básicos (LBP, Haralick, Gabor)

En primer lugar, se consideran los descriptores clásicos aplicados de forma individual:

- **LBP multiescala:** se emplean patrones binarios locales sobre imágenes en escala de grises, combinando varios radios y un número fijo de vecinos. Se utiliza una codificación basada en patrones uniformes que permite reducir la dimensionalidad del histograma manteniendo una buena capacidad descriptiva.
- **Características de Haralick / GLCM:** se cuantifican los niveles de gris de la imagen y se calculan matrices de coocurrencia para varias distancias y orientaciones. A partir de cada GLCM se extrae un conjunto de propiedades estadísticas (contraste, disimilitud, homogeneidad, energía, correlación, ASM).
- **Filtros de Gabor:** se diseña un banco de filtros de Gabor con un número limitado de frecuencias y orientaciones. La imagen se convoluciona con cada filtro del banco y las respuestas se resumen mediante estadísticas sencillas.

3.2.2. Bloque 2 – Embeddings de ResNet-50 (feature extraction congelado)

En un segundo bloque se sitúan los embeddings extraídos de una arquitectura ResNet-50 preentrenada en ImageNet. En esta configuración, la red se utiliza en modo feature extractor congelado:

- Se carga ResNet-50 con pesos preentrenados.
- Se congelan las capas convolucionales.
- Se sustituye la cabeza de clasificación final por una capa adaptada al número de clases del DTD (para entrenamiento end-to-end).

- Para la comparación “justa”, se toma como embedding la salida de la última capa convolucional tras aplicar global average pooling, obteniendo un vector de características de dimensión fija (≈ 2048 componentes).
- Este vector se normaliza y se emplea como entrada de la misma SVM utilizada para los descriptores clásicos.

3.2.3. Bloque 3 – Embeddings de ResNet-50 con fine-tuning

Un tercer bloque está formado por embeddings procedentes de ResNet-50 con fine-tuning parcial sobre el DTD:

- Se parte de la misma ResNet-50 preentrenada.
- Se descongelan las capas superiores (por ejemplo, el último bloque residual), manteniendo congeladas el resto.
- Se entrena la red con data augmentation sobre los splits de entrenamiento del DTD, usando una tasa de aprendizaje reducida para evitar destruir el conocimiento previo.
- Una vez entrenada la red, se extraen de nuevo los embeddings a partir del global average pooling de la última capa convolucional.
- Estos vectores se normalizan y se clasifican con la misma SVM.

3.3. Preparación de datos y particionado del DTD

La preparación de los datos constituye un elemento crítico para asegurar la comparabilidad entre métodos. El punto de partida es el Describable Textures Dataset (DTD). En este trabajo se siguen las particiones oficiales propuestas por los autores del dataset, que dividen las 5 640 imágenes en 10 splits independientes. Cada split separa, para cada una de las 47 clases, 40 imágenes para entrenamiento, 40 para validación y 40 para prueba.

3.4. Escenarios experimentales

Para responder de forma estructurada a las preguntas de investigación planteadas en la Introducción, el diseño experimental se articula en torno a cuatro escenarios

conceptuales:

- **Escenario A – Rendimiento en clasificación (núcleo del TFG):** Se evalúa el rendimiento de cada método en la tarea de clasificación de texturas sobre el DTD sin introducir perturbaciones adicionales.
- **Escenario B – Eficiencia computacional:** Este escenario se centra en comparar de forma aproximada la latencia de inferencia de cada pipeline.
- **Escenario C – Explicabilidad de las decisiones basadas en CNN:** En este escenario se aplica la técnica Grad-CAM sobre los modelos de ResNet-50 entrenados.

3.5. Criterios de evaluación y métricas

Con el fin de comparar de forma coherente los métodos considerados, se ha definido un conjunto reducido de métricas y criterios de evaluación comunes a todos ellos. El análisis cuantitativo se centra en tres aspectos principales: rendimiento en clasificación, estabilidad entre particiones del dataset, y eficiencia computacional. Adicionalmente, se incluye un análisis cualitativo de explicabilidad mediante mapas de activación Grad-CAM.

3.5.1. Métrica de rendimiento principal: *accuracy*

La métrica de rendimiento principal utilizada en este TFG es la *accuracy* clásica, definida, para cada método m y cada *split* s del DTD, como:

$$\text{acc}_{m,s} = \frac{1}{N_s} \sum_{i=1}^{N_s} \mathbb{I}[\hat{y}_i^{(m,s)} = y_i],$$

donde N_s es el número de imágenes de prueba en el *split* s , y_i es la etiqueta verdadera de la imagen i y $\hat{y}_i^{(m,s)}$ la etiqueta predicha por el método m .

Para cada método se calcula:

- La *accuracy media* sobre los diez *splits* oficiales del DTD:

$$\overline{\text{acc}}_m = \frac{1}{10} \sum_{s=1}^{10} \text{acc}_{m,s},$$

que representa el rendimiento global del método.

- La **desviación estándar** de la *accuracy* entre *splits*, que se utiliza como medida de *estabilidad* o variabilidad respecto al particionado:

$$\sigma(\text{acc}_m) = \sqrt{\frac{1}{9} \sum_{s=1}^{10} (\text{acc}_{m,s} - \overline{\text{acc}}_m)^2}.$$

Estas dos magnitudes (media y desviación estándar) son las que se emplean en el Capítulo 5 para construir las tablas y gráficas comparativas de rendimiento, y en el Capítulo 6 para discutir la estabilidad de los métodos.

Aunque existen otras métricas de clasificación multiclase (como el F1-score macro o la precisión por clase), en este TFG la *accuracy* se adopta como métrica principal al tratarse de un problema de clasificación balanceado con 47 clases y un número similar de muestras por categoría.

3.5.2. Análisis de errores: matrices de confusión

Para complementar la información agregada de la *accuracy*, se calculan matrices de confusión para cada método y *split*. Cada matriz resume, a nivel de clase, cuántas muestras de una determinada textura se han clasificado correctamente y en qué otras clases se concentran los errores.

En el análisis se utilizan:

- **Matrices de confusión por método**, agregadas a partir de los diez *splits* para obtener una visión global de los patrones de error.
- **Comparaciones cualitativas** entre familias de métodos (descriptores clásicos frente a configuraciones basadas en ResNet-50), identificando si las confusiones se distribuyen de forma uniforme o se concentran en determinadas parejas de clases visualmente similares.

Estas matrices se emplean de forma explícita en los Capítulos 5 y 6 para responder a las preguntas relacionadas con los patrones de confusión y el comportamiento diferencial de cada familia de representaciones.

3.5.3. Métricas de eficiencia computacional

Además del rendimiento en clasificación, se evalúa la eficiencia computacional de los distintos métodos mediante dos magnitudes sencillas:

- **Tiempo de entrenamiento por *split*** (*Tiempo Entr. (s)*), medido como el tiempo de reloj (wall-clock) transcurrido entre el inicio y el final del entrenamiento del modelo correspondiente a ese *split*. En el caso de las SVM, este tiempo incluye la búsqueda de hiperparámetros mediante validación cruzada; en el caso de las CNN, incluye todas las épocas de entrenamiento hasta obtener el mejor modelo según la métrica de validación.
- **Tiempo de inferencia por imagen** (*Tiempo Infer. (ms/img)*), estimado dividiendo el tiempo total empleado en clasificar un conjunto de prueba entre el número de imágenes procesadas, una vez descartada una breve fase de calentamiento (*warm-up*) de la GPU.

A partir de estos tiempos se discute, de forma cualitativa, la viabilidad de cada método en escenarios de automatización con restricciones de tiempo real y recursos, tal como se recoge en los Capítulos 5, 6 y 7.

3.5.4. Criterios de explicabilidad

Finalmente, para las configuraciones basadas en ResNet-50 se generan mapas de activación Grad-CAM sobre ejemplos representativos de distintas clases del DTD. En este caso no se define una métrica numérica específica, sino que se adopta un **criterio cualitativo**:

- Se considera deseable que las regiones de alta activación se correspondan con áreas de la imagen donde la textura objetivo es más prominente.
- Se analizan casos de acierto y de error, evaluando si las regiones destacadas por Grad-CAM son coherentes con la clase predicha.

Este análisis cualitativo se utiliza en los Capítulos 5, 6 y 7 para complementar las métricas cuantitativas y aportar información sobre cómo toman sus decisiones las arquitecturas profundas.

3.6. Protocolo experimental común

Con el objetivo de garantizar una comparación justa entre todos los métodos y facilitar la reproducibilidad, se ha seguido un protocolo experimental común basado en los diez *splits* oficiales del Describable Textures Dataset (DTD). Para cada método y cada *split* se aplica la siguiente estrategia general:

1. **Uso de los *splits* oficiales:** para cada $s \in \{1, \dots, 10\}$ se emplea la partición proporcionada por los autores del DTD, que divide el conjunto en imágenes de entrenamiento y de prueba. En el caso de las CNN, dentro de la partición de entrenamiento se reserva de forma sistemática una fracción para validación interna.
2. **Preprocesado común:**
 - Todas las imágenes se redimensionan a 224×224 píxeles.
 - Para los descriptores clásicos (LBP, GLCM, Gabor) se trabaja en escala de grises.
 - Para las arquitecturas basadas en ResNet-50 se utilizan imágenes en color normalizadas según las estadísticas habituales de ImageNet.
3. **Entrenamiento y evaluación por familia de métodos:**
 - **Descriptores clásicos + SVM:**
 - a) Extracción de descriptores (LBP multiescala, GLCM, Gabor) para todas las imágenes de entrenamiento y prueba del *split*, generando vectores de características que se almacenan en archivos comprimidos `.npz`.
 - b) Construcción de un *pipeline* de `scikit-learn` que incluye estandarización, PCA opcional y una SVM con kernel RBF.
 - c) Selección de hiperparámetros mediante búsqueda en rejilla (*Grid-SearchCV*) con validación cruzada estratificada sobre el conjunto de entrenamiento.
 - d) Entrenamiento final de la SVM con los mejores hiperparámetros encontrados y evaluación sobre el conjunto de prueba del *split*, almacenando las predicciones, la *accuracy* y la matriz de confusión.
 - **ResNet-50 end-to-end:**
 - a) Carga de ResNet-50 preentrenada en ImageNet y sustitución de la capa totalmente conectada final por una nueva capa adaptada a las 47 clases del DTD.

- b) Configuración del modo de entrenamiento (Frozen o con *fine-tuning* parcial) y definición del criterio de selección del mejor modelo en función del rendimiento sobre el subconjunto de validación.
- c) Entrenamiento durante un número fijo de épocas, con monitorización de la pérdida y la *accuracy* de validación, guardando el mejor modelo según dicha métrica.
- d) Evaluación del modelo seleccionado sobre el conjunto de prueba del *split*, registrando la *accuracy* y la matriz de confusión.

■ **ResNet-50 como extractor de características + SVM:**

- a) Carga del modelo ResNet-50 entrenado (congelado o con *fine-tuning*) y extracción de *embeddings* a partir de la última capa convolucional tras el *global average pooling*.
- b) Almacenamiento de los vectores de *embeddings* en archivos *.npz* para las imágenes de entrenamiento y prueba.
- c) Entrenamiento de una SVM con kernel RBF sobre los *embeddings* de entrenamiento, utilizando el mismo protocolo de búsqueda de hiperparámetros y validación cruzada que en el caso de los descriptores clásicos.
- d) Evaluación final sobre el conjunto de prueba, registrando *accuracy* y matriz de confusión.

4. **Agregación de resultados:** para cada método se agregan las métricas obtenidas en los diez *splits* (media y desviación estándar de la *accuracy*) y se generan tablas y gráficas comparativas. Este proceso de agregación se lleva a cabo mediante scripts específicos de análisis que recorren los archivos de resultados producidos por cada experimento.

Este protocolo asegura que todos los métodos se entrenan y evalúan bajo condiciones lo más comparables posible, compartiendo preprocesado, particiones de datos y criterios de selección de modelos.

3.7. Reproducibilidad y trazabilidad de los experimentos

Para facilitar la reproducibilidad del estudio y la trazabilidad de los resultados, se han adoptado varias medidas en la implementación software y en la organización de los experimentos:

- **Estructura modular del proyecto:** el código se organiza en módulos separados para el preprocesado del DTD, la extracción de descriptores clásicos, el entrenamiento de CNN, la extracción de *embeddings* y el entrenamiento del clasificador unificado (SVM), tal como se describe en el Capítulo 4. Esto permite reejecutar cada bloque de forma independiente.
- **Scripts parametrizables desde línea de comandos:** los scripts principales de entrenamiento y extracción de características aceptan argumentos que permiten seleccionar el método, el *split*, el modo de entrenamiento (Frozen/*fine-tuning*), las rutas de entrada y salida, etc. De este modo, la configuración de cada experimento queda explícita en la línea de comandos utilizada.
- **Registro sistemático de resultados:** cada ejecución genera archivos de salida con un identificador único (*run_id*) que codifica el método, el *split* y la fecha/hora. Los resultados cuantitativos (métricas, hiperparámetros seleccionados, etc.) se almacenan en archivos de texto estructurados (por ejemplo, *.csv* o *.json*), que se utilizan posteriormente en los scripts de análisis del Capítulo 5.
- **Control del entorno software:** se ha fijado un entorno virtual de Python específico para el proyecto y se ha documentado el conjunto de dependencias mediante un fichero *requirements.txt*, de forma que el entorno puede recrearse en otra máquina con versiones compatibles de las librerías.
- **Uso de semillas aleatorias:** cuando ha sido posible, se han fijado semillas aleatorias en las librerías principales (NumPy, PyTorch, *scikit-learn*) para reducir la variabilidad entre ejecuciones. No obstante, se asume que en entornos con GPU pueden existir pequeñas diferencias numéricas entre repeticiones.

Gracias a estas medidas, un tercero con acceso al código y al dataset DTD podría reproducir, al menos de forma aproximada, los resultados cuantitativos presentados en los capítulos de Resultados, Discusión y Respuestas a las Preguntas de Investigación, así como extender el estudio a nuevos métodos o configuraciones manteniendo el mismo marco experimental.

4. Desarrollo experimental

En este capítulo se detalla la implementación técnica del diseño metodológico descrito en el capítulo anterior. Se describe el entorno de desarrollo, la arquitectura software construida para orquestar los experimentos y los detalles de implementación de los pipelines de procesamiento, tanto para los descriptores clásicos como para las arquitecturas basadas en ResNet-50. El objetivo es proporcionar una visión clara de cómo se ha materializado el sistema para garantizar la reproducibilidad de los resultados.

4.1. Entorno hardware y software

El desarrollo y la ejecución de los experimentos se han llevado a cabo en un entorno de computación diseñado para soportar tanto el entrenamiento de modelos de aprendizaje profundo como el procesamiento masivo de descriptores de textura. A nivel de hardware, se ha utilizado una estación de trabajo con las siguientes especificaciones:

- **Procesador (CPU):** Intel Core i9-13900H (14 núcleos).
- **Memoria RAM:** 16 GB DDR5 a 5200 MT/s.
- **Acelerador gráfico (GPU):** NVIDIA GeForce RTX 4060 con 8 GB de VRAM y soporte CUDA.
- **Almacenamiento:** Unidad SSD, que permite una carga rápida del dataset DTD y de los modelos durante el entrenamiento.

4.2. Implementación del pipeline de descriptores clásicos

La implementación de los métodos clásicos se ha estructurado de forma modular, separando la lógica de carga de datos, preprocesado, extracción de características y clasificación. Los algoritmos de LBP, GLCM y Gabor se encapsulan en módulos específicos, mientras que el resto de la lógica se gestiona desde un script orquestador.

El flujo de procesamiento para este bloque es el siguiente:

- **Carga y preprocesado de imágenes:** Se ha implementado una clase `DTDTextureDataset` que lee los splits oficiales, carga las imágenes y las redimensiona a 224x224 píxeles.

- **Extracción de características:** Para cada imagen en escala de grises se aplica el descriptor correspondiente (LBP multiescala, Haralick/GLCM, Gabor).
- **Persistencia de embeddings clásicos:** Los vectores de características y etiquetas se guardan en archivos comprimidos `.npz`.
- **Estandarización previa a la SVM:** La estandarización (p. ej., `StandardScaler`) forma parte del pipeline de *scikit-learn*.

4.3. Implementación del pipeline basado en CNN

El pipeline de aprendizaje profundo se ha diseñado específicamente en torno a ResNet-50, utilizando exclusivamente esta arquitectura tanto en modo end-to-end como en modo extractor de características para la SVM.

4.3.1. Entrenamiento de ResNet-50

El subcomando `train_cnn` se encarga de:

- Cargar ResNet-50 preentrenada en ImageNet.
- Sustituir la capa totalmente conectada final por una capa adaptada a las 47 clases del DTD.
- Configurar el modo de entrenamiento (*Frozen* o *Fine-tuning*).
- Ejecutar el bucle de entrenamiento y guardar el mejor modelo por split.

4.3.2. Extracción de embeddings de ResNet-50

La extracción de embeddings se implementa mediante la carga del modelo entrenado y el registro de hooks para obtener la salida de la última capa convolucional tras el *global average pooling* (vector de aproximadamente 2048 dimensiones). Estos embeddings se guardan en archivos `.npz`.

4.4. Extracción de descriptores y entrenamiento de la SVM

El módulo `src/train_svm_embeddings.py` constituye el núcleo del esquema de “clasificador unificado”. Su cometido es cargar los archivos `.npz`, construir un pipeline

de *scikit-learn* (estandarización, PCA opcional, SVM con kernel RBF) y ejecutar una búsqueda en rejilla (*GridSearchCV*) con validación cruzada estratificada para seleccionar los mejores hiperparámetros.

Las métricas de cada *split* (precisión en entrenamiento, precisión en test y *gap* de sobreajuste) se vierten automáticamente a ficheros CSV y se recogen de forma detallada en el Anexo B.

4.5. Organización práctica de los experimentos

Para gestionar la complejidad combinatoria (múltiples descriptores, dos variantes de ResNet-50, 10 splits, etc.), se ha adoptado una estructura de proyecto modular y se ha automatizado la ejecución mediante scripts maestros.



5. Resultados

En este capítulo se presentan los datos cuantitativos obtenidos tras la ejecución de los experimentos definidos en la metodología. Se analiza el rendimiento de los descriptores clásicos (LBP, Gabor, GLCM) frente a las arquitecturas de aprendizaje profundo (basadas en ResNet-50) en el conjunto de datos de texturas (DTD), prestando especial atención a la capacidad de generalización y a la estabilidad del entrenamiento.

5.1. Rendimiento global y comparación de métodos

La Tabla 5.1 resume el rendimiento medio (*accuracy*) y la desviación típica obtenida por cada método a lo largo de los 10 *splits* oficiales del dataset DTD. Además, se incluye el rendimiento en el conjunto de entrenamiento para cuantificar el sobreajuste (*Gap*). Los valores individuales por *split* se detallan en el Anexo B.

Tabla 5.1: Resumen comparativo de rendimiento (promedio de 10 *splits*). El *Gap* representa la diferencia entre *accuracy* en entrenamiento y test, actuando como indicador de sobreajuste.

Método	Test Accuracy ($\mu \pm \sigma$)	Train Accuracy (μ)	Gap (Overfitting)	Inferencia (ms)
SVM + GLCM	0,195 $\pm$ 0,008	0.350	0.155	9.4
SVM + Gabor	0,218 $\pm$ 0,013	0.430	0.212	5.8
SVM + LBP	0,314 $\pm$ 0,012	0.645	0.331	2.1
ResNet-50 (Frozen) + SVM	0,586 $\pm$ 0,009	0.948	0.362	6.9
ResNet-50 (Finetune) + SVM	0.697 $\pm$ 0.012	0.947	0.250	6.9

Se observa una diferencia muy significativa entre las técnicas basadas en descriptores diseñados manualmente y las técnicas de *Deep Learning*. El mejor modelo clásico (LBP multiescala) alcanza un modesto 31.4% de acierto, mientras que la configuración basada en ResNet-50 con *fine-tuning* supera el 69.7%, duplicando sobradamente la capacidad discriminativa.

Es notable que el modelo **Frozen** (características fijas de ImageNet) obtiene un 58.6% de acierto con un *gap* de sobreajuste muy elevado (0.362), indicando que aunque la red memoriza el entrenamiento casi perfectamente (94.8%), las características genéricas no generalizan bien a texturas específicas sin adaptación. El **Fine-tuning** reduce este *gap* a 0.250 y eleva la precisión en test más de 10 puntos porcentuales, coherente con la evidencia previa sobre la necesidad de especializar las características profundas al dominio de interés [21], [24], [28].

5.2. Análisis de convergencia y entrenamiento

Para evaluar la estabilidad del proceso de optimización en las arquitecturas profundas, se analizaron las curvas agregadas de pérdida (*loss*) y precisión para los 10 *splits*. Estas curvas permiten estudiar no solo la convergencia del modelo, sino también el grado de generalización entre los datos de entrenamiento y validación.

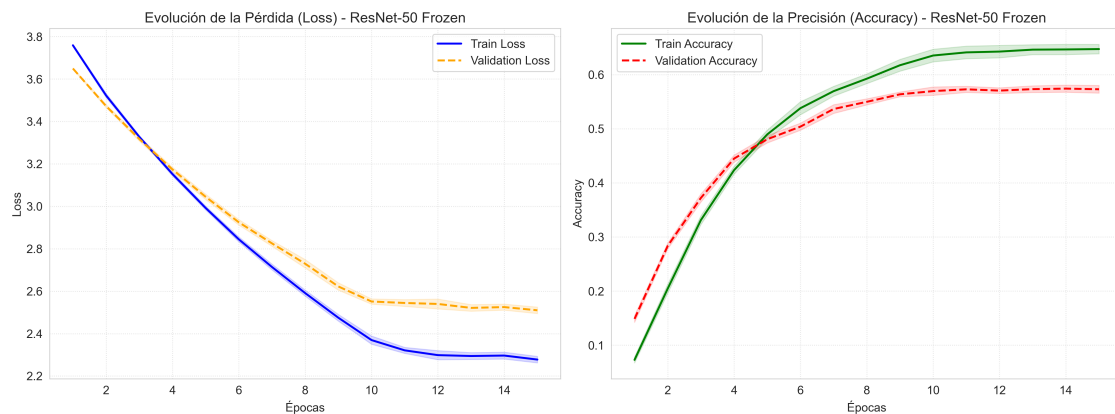


Figura 5.1: Curvas de aprendizaje agregadas para ResNet-50 *Frozen*. La sombra representa la desviación estándar entre *splits*. Se observa que la pérdida de validación se estabiliza tras las primeras épocas, mientras que la pérdida de entrenamiento continúa descendiendo, lo que indica que la adaptación de las características preentrenadas al dominio del DTD es limitada bajo este esquema.

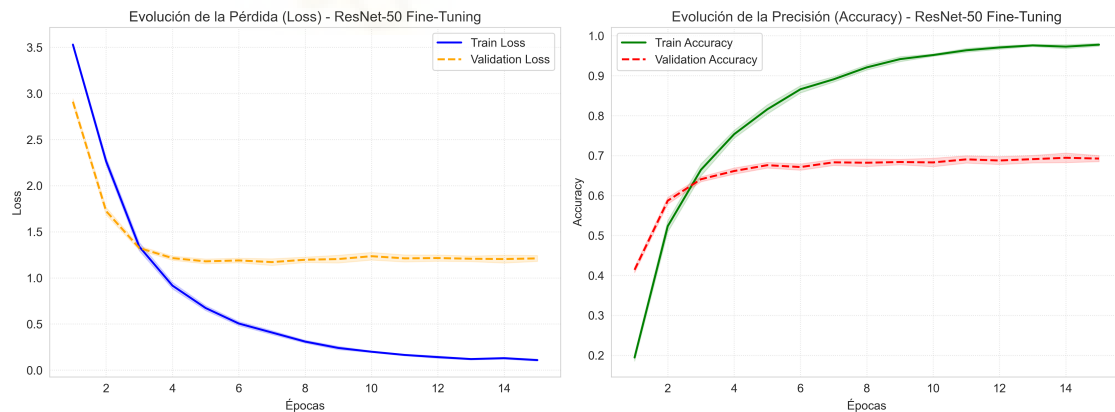


Figura 5.2: Curvas de aprendizaje agregadas para ResNet-50 *Fine-Tuning*. A diferencia del modelo congelado, el ajuste fino muestra una mejora progresiva tanto en pérdida como en precisión. Sin embargo, la separación entre las curvas de entrenamiento y validación tras las primeras épocas refleja signos de sobreajuste, pese al incremento en precisión final.

En la Figura 5.2 se aprecia que el *fine-tuning* permite una convergencia más prolongada y una mayor capacidad de adaptación al dominio, alcanzando precisiones

superiores respecto al modelo *Frozen*. No obstante, la divergencia entre entrenamiento y validación evidencia que el modelo desarrolla cierta memorización del conjunto de entrenamiento, lo que podría mitigarse mediante regularización adicional, *data augmentation* más agresivo o una selección alternativa de hiperparámetros.

La banda de desviación estándar en ambos casos es estrecha, lo que sugiere que el comportamiento del modelo es consistente entre particiones y que los resultados no dependen de un único *split*, reforzando la validez experimental del análisis.

5.3. Análisis de confusiones y errores

Para comprender cualitativamente la mejora de rendimiento, se comparan las matrices de confusión acumuladas y normalizadas del mejor método clásico frente al mejor método profundo.

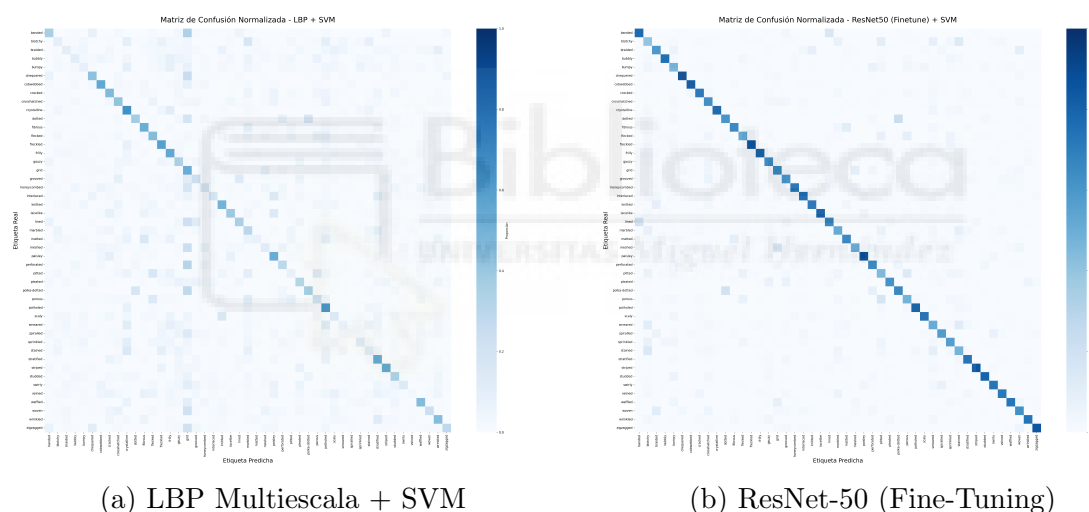


Figura 5.3: Comparativa de matrices de confusión normalizadas agregadas (10 *splits*). Se observa una diagonal significativamente más definida y limpia en el modelo de aprendizaje profundo (b), frente a la dispersión de ruido en el modelo clásico (a).

La Figura 5.3 ilustra cómo ResNet-50 logra concentrar la densidad de probabilidad en la diagonal principal. Mientras que LBP presenta un ruido de fondo generalizado (confusiones aleatorias entre clases no relacionadas), los errores de ResNet tienden a agruparse en bloques o pares específicos de texturas semánticamente similares (por ejemplo, confusión entre *braided* y *woven*), lo que indica que el modelo ha aprendido representaciones visuales coherentes.

5.4. Resultados de explicabilidad (Grad-CAM)

El análisis cualitativo mediante Grad-CAM confirma que la red extrae características de las regiones texturales relevantes [25].

Estos mapas (Figura 5.4) validan que el alto rendimiento cuantitativo (69.7 %) no se debe a sesgos del fondo o bordes de la imagen, sino a la detección efectiva de patrones como rejillas, burbujas o fibras.

Se observa una correlación directa entre la intensidad de activación de los gradientes y el rendimiento por clase observado previamente en las matrices de confusión (ver Figura 5.3b). En concreto:

- **Activación en patrones definidos:** Para las clases que presentan una mayor tasa de aciertos en la diagonal de la matriz de confusión (como *honeycombed*, *chequered* o *braided*), los mapas Grad-CAM muestran una activación intensa y focalizada sobre los elementos repetitivos de la textura. La red ignora eficazmente el fondo o la iluminación global, concentrándose en la estructura geométrica local (celdas, cuadros o trenzados).
- **Coherencia con la clasificación:** Esta focalización confirma que el alto rendimiento cuantitativo (accuracy del 69.7 %) no es producto del azar o de sesgos en los bordes de la imagen, sino que el modelo ha aprendido a caracterizar la topología de la textura de forma similar a como lo haría un experto humano.
- **Contraste con descriptores clásicos:** A diferencia de los descriptores como LBP, que mostraron confusión dispersa debido a su sensibilidad al ruido de píxel, la red neuronal demuestra mediante estos mapas de atención que es capaz de abstraer el concepto de “textura” por encima de las variaciones locales de intensidad.

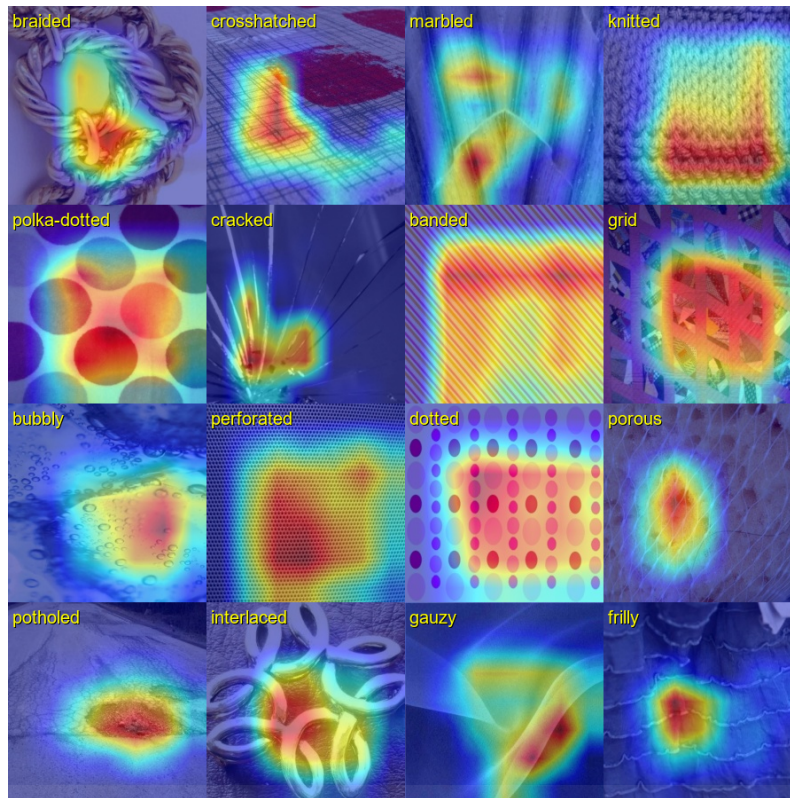


Figura 5.4: Mapas de activación Grad-CAM en ResNet-50. Las zonas cálidas indican las regiones que maximizan la clase predicha, coincidiendo con los patrones repetitivos de la textura.

6. Discusión

6.1. La brecha entre descriptores clásicos y profundos

Los resultados obtenidos ratifican la obsolescencia de los descriptores manuales (*hand-crafted*) para tareas de clasificación de texturas complejas “en el mundo real” como DTD. LBP, a pesar de ser el mejor de los clásicos con un 31.4 % de precisión, resulta claramente insuficiente para aplicaciones prácticas en este dominio. La incapacidad de LBP, Gabor y GLCM para modelar de forma conjunta variaciones de escala, iluminación y perspectiva limita severamente su rendimiento [7], [10], [20].

Por el contrario, ResNet-50 con *fine-tuning* logra duplicar el rendimiento de los descriptores clásicos. Esto demuestra que la jerarquía de características aprendidas (desde bordes simples hasta patrones complejos) es fundamental para desambiguar clases que comparten propiedades estadísticas de bajo nivel pero difieren en estructura semántica, tal y como ya se ha puesto de manifiesto en la literatura reciente sobre representaciones de textura [19].

Este comportamiento es coherente con el análisis original de Cimpoi et al., donde se introduce DTD y se observa que descriptores basados en LBP o bancos de filtros clásicos quedan claramente por detrás de combinaciones más ricas como SIFT+IFV o descriptores profundos (DeCAF) [3]. La revisión de Liu et al. muestra además que, en DTD, modelos basados en VGG con estrategias de codificación avanzadas (p. ej. VGG-VD + IFV o variantes bilineales) alcanzan alrededor de un 72–74 % de *accuracy* [19], mientras que arquitecturas *hand-crafted* como ScatNet apenas superan el 35 %. En este contexto, el 69.7 % obtenido con ResNet-50 *fine-tuning* y SVM se sitúa en la banda esperada: por debajo de las configuraciones más sofisticadas, pero claramente en la zona de eficacia propia de las CNN finamente ajustadas sobre DTD.

En resumen, la fuerte brecha observada entre descriptores clásicos y profundos no es un artefacto de la implementación, sino una manifestación más de la transición descrita en la literatura desde representaciones *BoW* y bancos de filtros hacia arquitecturas profundas como estándar de facto para texturas complejas [19].

6.2. Robustez y estabilidad del entrenamiento

Un hallazgo relevante del estudio es el análisis del **gap de generalización** (Tabla 5.1 y Anexo B):

- **Gabor + SVM** presenta una alta inestabilidad. Como se observa en los datos detallados por *split*, el *gap* oscila entre valores moderados ($\approx 0,11$) y muy altos (por encima de 0,35). Esta variabilidad es coherente con la sensibilidad de los filtros de Gabor a la orientación, frecuencia y escala de la textura: ante variaciones no controladas en el conjunto DTD, pequeñas diferencias en el muestreo pueden traducirse en grandes cambios en las respuestas de los filtros.
- **ResNet Frozen** muestra el mayor sobreajuste relativo (gap medio de 0.362). Al no permitir que la red adapte sus filtros internos al dominio de texturas, el clasificador final (SVM) memoriza las características extraídas en el conjunto de entrenamiento, pero estas no son lo suficientemente expresivas para generalizar bien a las imágenes de test.
- **ResNet Finetune** reduce este gap a 0.250. Ajustar los pesos permite a la red aprender invariancias específicas del dominio de texturas, mejorando la generalización. Este comportamiento coincide con los trabajos sobre *transfer learning* que muestran cómo el ajuste fino de capas profundas sobre el dominio objetivo mejora la transferencia frente al uso de características “off-the-shelf” [21], [24], [28].

En términos prácticos, los resultados indican que el modelo *Frozen* se comporta como un extractor de características genérico de ImageNet que no termina de adaptarse a las texturas “en la naturaleza” (DTD), mientras que el modelo *Finetune* sacrifica algo de estabilidad en el entrenamiento (mayor riesgo de sobreajuste) a cambio de una ganancia clara en rendimiento y coherencia semántica de los errores.

6.3. Eficiencia vs. precisión: implicaciones prácticas

Desde la perspectiva de la ingeniería de automatización, la elección del modelo plantea un compromiso claro:

1. **Escenarios de bajos recursos (CPU / Edge):** Si el hardware es muy restrictivo, LBP es la opción más ligera (2.1 ms por imagen) pero su baja fiabilidad (31 % de acierto) lo descarta para aplicaciones de control de calidad crítico. Esta conclusión es consistente con los resultados reportados en la literatura: en bases de datos controladas como Outex, LBP y variantes modernas pueden alcanzar rendimientos casi perfectos, pero en texturas “in the wild” como DTD su rendimiento se degrada notablemente [19], [20].

2. **Escenarios de alto rendimiento (GPU):** ResNet-50 ofrece una precisión del 70 % con tiempos de inferencia de 6.9 ms. Este tiempo es perfectamente compatible con líneas de producción estándar (que operan a ritmos de 10–20 piezas por segundo), justificando el uso de hardware acelerador por la ganancia en calidad de inspección. Además, la literatura muestra que arquitecturas diseñadas específicamente para eficiencia (MobileNet, EfficientNet, etc.) pueden reducir aún más el coste computacional manteniendo niveles de precisión comparables [27], lo que abre la puerta a implementaciones embebidas más exigentes.

En conjunto, los resultados apoyan la idea de que, para escenarios industriales exigentes, el coste adicional de desplegar una CNN moderna frente a un descriptor clásico está ampliamente justificado por la mejora en precisión y la coherencia semántica de los errores.

7. Conclusiones y respuestas a las preguntas de investigación

El presente trabajo ha evaluado comparativamente la eficacia de descriptores clásicos y arquitecturas profundas para la clasificación de texturas complejas. A continuación se da respuesta a las preguntas planteadas al inicio del estudio:

7.1. ¿Superan las CNN a los descriptores clásicos en rendimiento?

Sí, de manera rotunda. Los resultados muestran que ResNet-50 con *fine-tuning* alcanza una precisión media del **69.7 %**, frente al **31.4 %** del mejor descriptor clásico (LBP). Esto supone una mejora relativa superior al 100 %, en línea con lo observado en la literatura para datasets de texturas en entornos no controlados como DTD [3], [19], y valida el uso de *Deep Learning* incluso en datasets de tamaño moderado.

7.2. ¿Cómo se comportan frente a los errores?

Mientras que los métodos clásicos presentan una matriz de confusión con ruido disperso (fallos aleatorios), las arquitecturas profundas concentran sus errores en

pares de clases visualmente ambiguas. Esto sugiere que los fallos de la CNN son “más humanos” y semánticamente coherentes, lo que facilita el diagnóstico de errores en un entorno industrial (por ejemplo, texturas muy similares que podrían ser aceptables o no según el contexto de la aplicación).

7.3. ¿Es viable computacionalmente en automatización?

Los tiempos de inferencia obtenidos (≈ 7 ms para ResNet-50 en GPU) son compatibles con la mayoría de aplicaciones de inspección industrial en tiempo real. Aunque el entrenamiento es costoso (minutos frente a segundos en el caso de los descriptores clásicos), la fase crítica en producción es la inferencia, y en este aspecto el coste adicional es asumible. La literatura reciente sugiere, además, que arquitecturas más eficientes podrían reducir aún más ese coste sin perder demasiada precisión [27].

7.4. ¿Qué aporta la explicabilidad?

Las visualizaciones Grad-CAM han demostrado que la red neuronal basa sus decisiones en patrones texturales legítimos y no en artefactos de fondo [25]. Esto aporta la confianza necesaria para desplegar estos modelos “caja negra” en entornos de ingeniería donde la trazabilidad es clave, ya que permite verificar que las regiones de activación coinciden con las zonas que un experto humano consideraría relevantes para la clasificación.

Más allá del análisis global de la Figura 5.4, el comportamiento de Grad-CAM resulta especialmente ilustrativo cuando se examinan clases concretas. En general, se observa que aquellas clases que presentan un mayor rendimiento en las matrices de confusión (Figura 5.3b) tienden a mostrar mapas de activación más focalizados en las zonas característicamente texturales de la imagen, mientras que las clases con peor rendimiento exhiben activaciones más difusas o desplazadas hacia regiones menos informativas.

Un ejemplo representativo es la clase *cobwebbed*, donde el modelo alcanza una alta tasa de acierto. Como se aprecia en la Figura 7.1, las regiones de máxima activación se concentran precisamente en la estructura entrecruzada que define la textura, ignorando en gran medida el fondo y otros detalles irrelevantes. Esta correspondencia entre zonas activas y patrón geométrico dominante respalda la idea de que la red ha aprendido a identificar el rasgo discriminativo clave de la clase, y no meros artefactos de iluminación o contorno. Desde el punto de vista de la ingeniería, este

tipo de visualización aporta evidencia adicional de que las decisiones del modelo se apoyan en atributos visuales razonables y facilita la validación del sistema en entornos industriales donde la trazabilidad es crítica.

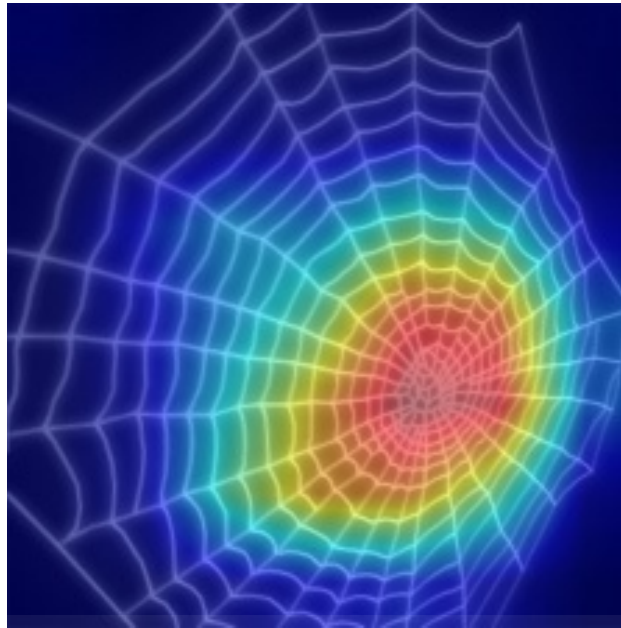


Figura 7.1: Mapa Grad-CAM para un ejemplo de la clase *cobwebbed*. Las zonas cálidas se alinean con la estructura de hilos entrecruzados típica de esta textura, indicando que la red fundamenta su decisión en el patrón textural relevante.

8. Trabajo futuro

A partir de los hallazgos y limitaciones de este estudio, se proponen las siguientes líneas de trabajo:

- **Aumento de datos (*Data Augmentation*) avanzado:** Dado el *gap* de sobreajuste observado (0.250) en el modelo *Finetune*, se propone explorar técnicas de aumento de datos más agresivas (rotaciones aleatorias, recortes, cambios de color) para forzar a la red a generalizar mejor.
- **Arquitecturas eficientes (MobileNet/EfficientNet):** Evaluar arquitecturas diseñadas para dispositivos móviles permitiría reducir los tiempos de inferencia y desplegar soluciones de *Deep Learning* en hardware embebido (Raspberry Pi, Jetson Nano) sin sacrificar excesiva precisión [27].
- **Fusión de características:** Investigar si la concatenación de descriptores LBP (que capturan micro-textura) con embeddings de CNN (que capturan estructura global) podría mejorar el rendimiento final, combinando lo mejor de ambos mundos [19].
- **Robustez experimental sistematizada:** En este trabajo no se ha abordado de forma sistemática la robustez de los modelos frente a perturbaciones (ruido, oclusiones, cambios de iluminación, rotaciones extremas, etc.). Como línea de trabajo futuro, se propone diseñar un protocolo experimental específico para evaluar y comparar la robustez de descriptores clásicos y CNN bajo distintos escenarios de degradación controlada, siguiendo enfoques similares a los descritos en la literatura reciente de reconocimiento robusto de texturas [19].

A. Código fuente y desarrollo software

El desarrollo experimental se ha estructurado como una *suite* de Python modular que permite la ejecución reproducible de todos los escenarios planteados. A continuación se presentan los módulos principales que constituyen el núcleo del Trabajo de Fin de Grado.

A.1. Orquestación y reproducibilidad

El control de los experimentos se realiza mediante un script maestro que define la configuración de los escenarios (clásicos vs. *deep*) y lanza los procesos de extracción y entrenamiento de forma secuencial.

```
1 # run_all_experiments.py
2 # Configuración de la batería de experimentos
3 CLASSIC_EXPERIMENTS: Dict[str, List[str]] = {
4     "classic_lbp": ["lbp"],
5     "classic_glcm": ["glcm"],
6     "classic_gabor": ["gabor"],
7     "classic_lbp_glcm_gabor": ["lbp", "glcm", "gabor"],
8 }
9
10 DEEP_EXPERIMENTS = [
11     {
12         "experiment_name": "deep_resnet50_frozen",
13         "model_name": "resnet50",
14         "finetune": False,
15     },
16     {
17         "experiment_name": "deep_resnet50_finetune",
18         "model_name": "resnet50",
19         "finetune": True,
20     },
21 ]
22
23 def run_deep_suite(splits, cv_folds, n_jobs, search_mode):
24     """
25     Automatiza el ciclo completo para Deep Learning:
26     1. Entrenar CNN (Frozen o Fine-tuning).
27     2. Extraer embeddings del modelo resultante.
28     3. Entrenar SVM sobre los embeddings.
29     """
30     for cfg in DEEP_EXPERIMENTS:
```

```

31     # ... (Lógica de iteración por splits y llamadas a
        subprocessos) ...
32     _run_train_cnn_cli(...)
33     _run_extract_embeddings_cli(...)
34     _run_svm_for_deep_experiment(...)

```

Listing 1: Script maestro de ejecución de experimentos (run_all_experiments.py)

```

1  # main.py
2  def build_main_parser() -> argparse.ArgumentParser:
3      """
4      Define los subcomandos disponibles en la suite.
5      """
6      parser = argparse.ArgumentParser(description="TFG Suite")
7      subparsers = parser.add_subparsers(dest="command", required=
        True)
8
9      # Subcomandos para cada etapa del pipeline
10     subparsers.add_parser("train_cnn", help="Entrenar CNN")
11     subparsers.add_parser("extract_embeddings", help="Extraer
        vectores")
12     subparsers.add_parser("train_svm_embeddings", help="Entrenar
        Clasificador Unificado")
13     subparsers.add_parser("classic_features", help="Pipeline de
        descriptores clásicos")
14
15     return parser

```

Listing 2: Punto de entrada único (main.py)

A.2. Implementación de descriptores clásicos

Se han implementado clases compatibles con la interfaz de *scikit-learn* para los descriptores de textura, permitiendo su integración en pipelines de procesamiento.

```

1  @dataclass
2  class MultiScaleLBP:
3      """
4      Extractor de LBP multi-escala con normalización de histogramas.
5      Combina múltiples radios para capturar texturas a diferentes
        resoluciones.
6      """
7      radii: Sequence[int] = (1, 3, 5)
8      method: str = "uniform"
9      normalize_hist: bool = True
10

```

```
11 def extract_single(self, img: np.ndarray) -> np.ndarray:
12     img_proc = self._preprocess_image(img)
13     feats = []
14     # Iterar sobre cada radio definido (multiescala)
15     for radius, P in zip(self.radii, self.num_points):
16         lbp_im = local_binary_pattern(img_proc, P, radius,
17                                     method=self.method)
18         hist = self._lbp_histogram(lbp_im, P)
19         feats.append(hist)
20
21     # Concatenar histogramas de todos los radios
22     return np.concatenate(feats, axis=0)
```

Listing 3: Descriptor LBP multiescala (lbp.py)

```
1 def compute_gabor_multiscale(
2     image: ArrayLike,
3     frequencies: Sequence[float] = (0.1, 0.2, 0.3, 0.4),
4     thetas: Optional[Sequence[float]] = None,
5 ) -> np.ndarray:
6     """
7     Aplica un banco de filtros de Gabor y extrae estadísticas (
8         media, std).
9     """
10    img = _to_grayscale(image)
11    feats: List[float] = []
12
13    for freq in frequencies:
14        for theta in thetas:
15            real, imag = gabor(img, frequency=freq, theta=theta)
16            magnitude = np.sqrt(real ** 2 + imag ** 2)
17
18            # Resumen estadístico de la respuesta del filtro
19            feats.append(float(np.mean(magnitude)))
20            feats.append(float(np.std(magnitude)))
21
22    return np.asarray(feats, dtype=np.float32)
```

Listing 4: Descriptor de filtros de Gabor (gabor.py)

A.3. Arquitectura y entrenamiento de modelos *Deep Learning*

Para garantizar una comparación justa, se han adaptado las arquitecturas SOTA eliminando sus cabezales de clasificación originales y permitiendo la extracción directa

de *embeddings*.

```

1 class CNNBackbone(nn.Module):
2     """
3     Wrapper para ResNet-50 eliminando la capa Fully Connected de
4     ImageNet.
5     Permite modos 'frozen' (extractor fijo) y 'finetune'.
6     """
7     def __init__(self, cfg: CNNBackboneConfig) -> None:
8         super().__init__()
9         if cfg.architecture == "resnet50":
10             backbone = resnet50(weights=ResNet50_Weights.DEFAULT)
11             self.embedding_dim = backbone.fc.in_features
12             # Sustitución de la cabeza de clasificación por
13             # Identidad
14             backbone.fc = nn.Identity()
15
16             self.backbone = backbone
17
18             # Nueva cabeza de clasificación adaptada al DTD (47 clases)
19             self.classifier = nn.Sequential(
20                 nn.Dropout(p=cfg.dropout),
21                 nn.Linear(self.embedding_dim, cfg.num_classes),
22             )
23
24     def forward_features(self, x: torch.Tensor) -> torch.Tensor:
25         """Devuelve el vector de características (embedding) antes
26         de clasificar."""
27         return self.backbone(x)

```

Listing 5: Adaptación de arquitecturas SOTA (cnn_backbones.py)

```

1 def train_cnn_main(args):
2     # ... Configuración de loaders y data augmentation ...
3
4     # Data augmentation ocurre en GPU para eficiencia
5     gpu_aug_train = GPUAugmentationPipeline(prepare_cfg, is_train=True)
6     .to(DEVICE)
7
8     for epoch in range(1, args.epochs + 1):
9         # Entrenamiento de una época
10         train_loss, train_acc, _ = train_one_epoch(
11             model, loaders["train"], criterion, optimizer, DEVICE,
12             gpu_aug_train
13         )
14
15         # Validación y guardado del mejor modelo (Checkpointing)
16         if val_acc > best_val_acc:

```

```
15         best_val_acc = val_acc
16         torch.save(model.state_dict(), best_model_path)
```

Listing 6: Bucle de entrenamiento (train_cnn.py)

A.4. Pipeline de clasificación unificado (SVM)

Este módulo es crítico para la metodología del trabajo, ya que asegura que tanto los descriptores clásicos como los *embeddings* de CNN sean evaluados por el mismo clasificador (SVM con kernel RBF) bajo las mismas condiciones de optimización.

```
1 def _build_svm_pipeline_and_grid(n_features, search_mode="balanced"
2 ):
3     """
4     Construye el pipeline: Estandarización -> (PCA opcional) -> SVM
5     RBF.
6     Define el espacio de búsqueda para C y Gamma.
7     """
8     steps = [("scaler", StandardScaler())]
9
10    # PCA automático si la dimensionalidad es alta (ej. ResNet =
11    2048 dims)
12    if n_features > 256:
13        steps.append(("pca", PCA()))
14
15    steps.append(("svm", SVC(kernel="rbf")))
16    pipeline = Pipeline(steps=steps)
17
18    # Definición del Grid de búsqueda logarítmica
19    param_grid = {
20        "svm__C": [2.0**p for p in [-4, -2, 0, 2]],
21        "svm__gamma": [2.0**g for g in [-8, -6, -4, -2]]
22    }
23    return pipeline, param_grid
24
25 def train_svm_embeddings_main(args):
26     # Carga de embeddings precomputados
27     X_trainval, y_trainval, X_test, y_test = _load_embeddings(...)
28
29     # Ejecución de GridSearchCV con validación cruzada
30     grid = GridSearchCV(
31         estimator=pipeline,
32         param_grid=param_grid,
33         cv=args.cv_folds,
34         scoring="accuracy",
```

```

32     n_jobs=-1
33 )
34 grid.fit(X_trainval, y_trainval)
35
36 # Evaluación final en Test
37 best_model = grid.best_estimator_
38 acc_test = accuracy_score(y_test, best_model.predict(X_test))

```

Listing 7: Entrenamiento de SVM con búsqueda de hiperparámetros (train_svm_embeddings.py)

```

1 @torch.no_grad()
2 def extract_embeddings_for_loader(model, loader, device):
3     """
4     Pasa el dataset por la CNN congelada y extrae los vectores
5     de la penúltima capa para usarlos en la SVM.
6     """
7     model.eval()
8     all_features = []
9
10    for images, labels in loader:
11        images = images.to(device)
12        # Se usa forward_features para obtener solo el embedding
13        features = model.forward_features(images)
14        all_features.append(features.cpu().numpy())
15
16    return np.concatenate(all_features, axis=0)

```

Listing 8: Extracción de embeddings para la SVM (extract_embeddings.py)

B. Métricas detalladas de los experimentos

En este anexo se recogen las métricas detalladas por *split* para cada uno de los métodos evaluados. Se incluyen, para cada *split*, la precisión en entrenamiento, la precisión en test y el *gap* de sobreajuste definido como:

$$\text{Gap} = \text{Train Accuracy} - \text{Test Accuracy}.$$

B.1. Descriptores clásicos + SVM

Tabla B.1: Métricas detalladas por *split* para GLCM + SVM.

Split	Train Acc.	Test Acc.	Gap
1	0.344	0.196	0.148
2	0.332	0.209	0.123
3	0.360	0.187	0.173
4	0.340	0.206	0.134
5	0.348	0.192	0.156
6	0.348	0.199	0.149
7	0.355	0.195	0.160
8	0.353	0.183	0.170
9	0.351	0.187	0.165
10	0.358	0.195	0.163

Tabla B.2: Métricas detalladas por *split* para Gabor + SVM.

Split	Train Acc.	Test Acc.	Gap
1	0.579	0.206	0.373
2	0.360	0.232	0.127
3	0.371	0.215	0.156
4	0.355	0.246	0.109
5	0.578	0.222	0.355
6	0.367	0.220	0.147
7	0.587	0.212	0.374
8	0.361	0.211	0.150
9	0.370	0.211	0.159
10	0.371	0.202	0.169

Tabla B.3: Métricas detalladas por *split* para LBP multiescala + SVM.

Split	Train Acc.	Test Acc.	Gap
1	0.660	0.314	0.346
2	0.645	0.335	0.310
3	0.669	0.307	0.361
4	0.649	0.321	0.328
5	0.660	0.309	0.351
6	0.665	0.321	0.344
7	0.668	0.321	0.347
8	0.668	0.290	0.378
9	0.437	0.307	0.129
10	0.669	0.310	0.359

B.2. Embeddings de ResNet-50 + SVM

Tabla B.4: Métricas detalladas por *split* para ResNet50 (Frozen) + SVM.

Split	Train Acc.	Test Acc.	Gap
1	0.949	0.591	0.358
2	0.949	0.607	0.341
3	0.944	0.576	0.368
4	0.949	0.587	0.363
5	0.944	0.586	0.358
6	0.951	0.590	0.361
7	0.952	0.579	0.373
8	0.947	0.578	0.369
9	0.948	0.584	0.364
10	0.949	0.582	0.367

Tabla B.5: Métricas detalladas por *split* para ResNet50 (Finetune) + SVM.

Split	Train Acc.	Test Acc.	Gap
1	0.948	0.706	0.241
2	0.949	0.718	0.231
3	0.945	0.687	0.259
4	0.908	0.679	0.228
5	0.951	0.707	0.244
6	0.949	0.698	0.252
7	0.944	0.686	0.258
8	0.953	0.695	0.258
9	0.947	0.685	0.263
10	0.945	0.688	0.257

B.3. ResNet-50 End-to-End (Softmax)

Estas tablas recogen los resultados obtenidos utilizando la arquitectura ResNet-50 completa con su clasificador lineal original (Softmax), sin emplear SVM.

Tabla B.6: Métricas detalladas por *split* para ResNet50 (Frozen) End-to-End.

Split	Train Acc.	Test Acc.	Gap
1	0.651	0.579	0.072
2	0.632	0.564	0.068
3	0.652	0.573	0.079
4	0.645	0.582	0.063
5	0.644	0.567	0.077
6	0.649	0.579	0.070
7	0.638	0.571	0.067
8	0.660	0.582	0.078
9	0.636	0.563	0.073
10	0.656	0.577	0.079

Tabla B.7: Métricas detalladas por *split* para ResNet50 (Finetune) End-to-End.

Split	Train Acc.	Test Acc.	Gap
1	0.977	0.712	0.265
2	0.975	0.687	0.288
3	0.977	0.701	0.276
4	0.977	0.676	0.301
5	0.980	0.687	0.293
6	0.981	0.695	0.286
7	0.975	0.701	0.274
8	0.981	0.696	0.285
9	0.977	0.698	0.279
10	0.980	0.692	0.288

s B.1–B.7 permiten reproducir los valores medios reportados y sirven como referencia complementaria para el análisis de los métodos de aprendizaje profundo.

BIBLIOGRAFÍA

- [1] B. E. Boser, I. M. Guyon y V. N. Vapnik, «A training algorithm for optimal margin classifiers,» en *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, 1992, págs. 144-152.
- [2] L. Breiman, «Random forests,» *Machine Learning*, vol. 45, n.º 1, págs. 5-32, 2001.
- [3] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed y A. Vedaldi, «Describing textures in the wild,» en *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, págs. 3606-3613.
- [4] M. Cimpoi, S. Maji y A. Vedaldi, «Deep filter banks for texture recognition and segmentation,» en *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, págs. 3828-3836.
- [5] C. Cortes y V. Vapnik, «Support-vector networks,» *Machine Learning*, vol. 20, n.º 3, págs. 273-297, 1995.
- [6] T. Cover y P. Hart, «Nearest neighbor pattern classification,» *IEEE Transactions on Information Theory*, vol. 13, n.º 1, págs. 21-27, 1967.
- [7] D. Gabor, «Theory of communication,» *Journal of the Institution of Electrical Engineers-Part III: Radio and Communication Engineering*, vol. 93, n.º 26, págs. 429-457, 1946.
- [8] R. C. Gonzalez y R. E. Woods, *Digital Image Processing*, 4.^a ed. Pearson, 2018.
- [9] I. Goodfellow, Y. Bengio y A. Courville, *Deep Learning*. MIT Press, 2016, <http://www.deeplearningbook.org>.
- [10] R. M. Haralick, K. Shanmugam e I. Dinstein, «Textural features for image classification,» *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 3, n.º 6, págs. 610-621, 1973.
- [11] T. Hastie, R. Tibshirani y J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2.^a ed. Springer, 2009.

- [12] K. He, X. Zhang, S. Ren y J. Sun, «Deep residual learning for image recognition,» en *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, págs. 770-778.
- [13] G. Huang, Z. Liu, L. Van Der Maaten y K. Q. Weinberger, «Densely connected convolutional networks,» en *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, págs. 4700-4708.
- [14] A. K. Jain, R. P. W. Duin y J. Mao, «Statistical pattern recognition: A review,» *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, n.º 1, págs. 4-37, 2000.
- [15] A. K. Jain y F. Farrokhnia, «Unsupervised texture segmentation using Gabor filters,» *Pattern Recognition*, vol. 24, n.º 12, págs. 1167-1186, 1991.
- [16] D. P. Kingma y J. Ba, «Adam: A Method for Stochastic Optimization,» en *International Conference on Learning Representations (ICLR)*, 2015.
- [17] A. Krizhevsky, I. Sutskever y G. E. Hinton, «ImageNet classification with deep convolutional neural networks,» en *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 25, 2012, págs. 1097-1105.
- [18] Y. LeCun, Y. Bengio y G. Hinton, «Deep learning,» *Nature*, vol. 521, n.º 7553, págs. 436-444, 2015.
- [19] L. Liu, P. Fieguth, Y. Guo, X. Wang, M. Pietikäinen y D. Hu, «From BoW to CNN: Two decades of texture representation for texture classification,» *International Journal of Computer Vision*, vol. 127, n.º 1, págs. 74-109, 2019.
- [20] T. Ojala, M. Pietikäinen y T. Mäenpää, «Multiresolution gray-scale and rotation invariant texture classification with local binary patterns,» *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, n.º 7, págs. 971-987, 2002.
- [21] S. J. Pan y Q. Yang, «A survey on transfer learning,» *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, n.º 10, págs. 1345-1359, 2010.
- [22] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga et al., «PyTorch: An Imperative Style,

High-Performance Deep Learning Library,» en *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, 2019.

- [23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot y E. Duchesnay, «Scikit-learn: Machine Learning in Python,» *Journal of Machine Learning Research*, vol. 12, págs. 2825-2830, 2011.
- [24] A. S. Razavian, H. Azizpour, J. Sullivan y S. Carlsson, «CNN features off-the-shelf: an astounding baseline for recognition,» en *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPR-W)*, 2014, págs. 806-813.
- [25] 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,» en *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017, págs. 618-626.
- [26] K. Simonyan y A. Zisserman, «Very deep convolutional networks for large-scale image recognition,» en *International Conference on Learning Representations (ICLR)*, 2015.
- [27] M. Tan y Q. Le, «EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks,» en *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019, págs. 6105-6114.
- [28] J. Yosinski, J. Clune, Y. Bengio y H. Lipson, «How transferable are features in deep neural networks?» En *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 27, 2014, págs. 3320-3328.