

INSTITUTO TECNOLÓGICO DE CHIHUAHUA
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

***“METODOLOGÍA PARA LA COMPRESIÓN DE
REDES NEURONALES PROFUNDAS
MEDIANTE TÉCNICAS DE INTELIGENCIA
ARTIFICIAL EXPLICABLE”***

TESIS

QUE PARA OBTENER EL GRADO DE

***MAESTRO EN CIENCIAS
EN INGENIERÍA ELECTRÓNICA***

PRESENTA:

AXEL FERNANDO TAM RUIZ

**DIRECTOR DE LA TESIS:
DR. JUAN ALBERTO RAMÍREZ QUINTANA**

CHIHUAHUA, CHIH., SEPTIEMBRE 2025



SEP
SECRETARÍA DE
EDUCACIÓN PÚBLICA



TECNOLÓGICO NACIONAL DE MÉXICO





Educación
Secretaría de Educación Pública



TECNOLÓGICO
NACIONAL DE MÉXICO



Instituto Tecnológico de Chihuahua
División de Estudios de Posgrado e Investigación

Chihuahua, Chih. 26 de septiembre de 2025

C. AXEL FERNANDO TAM RUIZ
PRESENTE

En cumplimiento con los requerimientos para la obtención del grado de Maestro en Ciencias en Ingeniería Electrónica y a propuesta de su Comité Tutorial, la División de Estudios de Posgrado e Investigación le concede la autorización para imprimir la tesis titulada **"Metodología para la comprensión de redes neuronales profundas mediante técnicas de inteligencia artificial explicable"** dirigida por Dr. Juan Alberto Ramírez Quintana con el siguiente contenido de capítulos:

CAPÍTULO I. MARCO TEÓRICO

CAPÍTULO II. ANTECEDENTES

CAPÍTULO III. ENTORNO DE PRUEBAS

CAPÍTULO IV. COMPARACIÓN DE MÉTODOS DE PODA DE REDES NEURONALES

CAPÍTULO V. EVALUACIÓN DE TÉCNICAS DE XAI PARA LA PODA ESTRUCTURAL

CAPÍTULO VI. METODOLOGÍA PROPUESTA DE PODA ESTRUCTURAL BASADA EN XAI

CAPÍTULO VII. RESULTADOS Y CONCLUSIONES

ATENTAMENTE

Excelencia en Educación Tecnológica
"La Técnica por el Engrandecimiento de México"

Leoncio



LEONCIO OCHOA CERVANTES

JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

LOC/vgo



2025
Año de
La Mujer
Indígena

Av. Tecnológico #2909 Col. 10 de Mayo
C.P. 31200, Chihuahua, Chihuahua. Tel. (614) 2012000 ext. 2157
e-mail: dir_chihuahua@tecnm.mx <http://itichihuahua.mx>





Educación
Secretaría de Educación Pública



TECNOLÓGICO
NACIONAL DE MÉXICO



Instituto Tecnológico de Chihuahua
División de Estudios de Posgrado e Investigación

Chihuahua, Chih. 23 de septiembre de 2025

LEONCIO OCHOA CERVANTES
JEFE DE LA DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN
PRESENTE

En cumplimiento con los requerimientos para la obtención del grado de Maestro en Ciencias en Ingeniería Electrónica, le notificamos que el documento de tesis del alumno C. AXEL FERNANDO TAM RUIZ, titulado *"Metodología para la comprensión de redes neuronales profundas mediante técnicas de inteligencia artificial explicable"* dirigido por Dr. Juan Alberto Ramírez Quintana, ha sido aprobado y aceptado para su impresión.

Por lo anterior, proponemos le sea concedida la autorización de impresión correspondiente.

Agradeciendo la atención a la presente, quedamos de usted.

ATENTAMENTE
Excellencia en Educación Tecnológica
La técnica por el Engendramiento de México

Juan Ramírez

DR. JUAN ALBERTO RAMÍREZ QUINTANA
MIEMBRO DEL COMITÉ TUTORIAL

Abimael G

DR. ABIMAEEL GUZMÁN PANDO
MIEMBRO DEL COMITÉ TUTORIAL

DR. JESÚS ALFONSO MEDRANO HERMOSILLO
MIEMBRO DEL COMITÉ TUTORIAL

DR. LUIS FRANCISCO CORRAL MARTÍNEZ
MIEMBRO DEL COMITÉ TUTORIAL



Av. Tecnológica 4000-Quinto de Mayo
C.P. 31004, Chihuahua, Chihuahua, Tel. (614) 2612000 ext. 2187
e-mail: dr_chihuahua@tecn.mx <http://itcdhihuahua.mx>





Educación
Secretaría de Educación Pública



TECNOLÓGICO
NACIONAL DE MÉXICO



Instituto Tecnológico de Chihuahua
División de Estudios de Posgrado e Investigación

CARTA CESIÓN DE DERECHOS

En la ciudad de Chihuahua el día 2 de octubre de 2025, el que suscribe, C. AXEL FERNANDO TAM RUIZ con número de control G23061082, de la Maestría en Ciencias en Ingeniería Electrónica, adscrita a la División de Estudios de Posgrado e Investigación del Instituto Tecnológico de Chihuahua, manifiesta que es autor intelectual del presente trabajo de Tesis bajo la dirección del Dr. Juan Alberto Ramírez Quintana y cede los derechos del trabajo titulado "Metodología para la compresión de redes neuronales profundas mediante técnicas de inteligencia artificial explicable", al Tecnológico Nacional de México y/o Instituto Tecnológico de Chihuahua para su difusión, divulgación, transmisión, reproducción, así como su digitalización con fines académicos y de investigación.

C. AXEL FERNANDO TAM RUIZ



Av. Tecnológico #2009 Col. 10 de Mayo
C.P. 31200, Chihuahua, Chihuahua. Tel. (614) 2012000 ext. 2157
e-mail: dir_chihuahua@tecnm.mx [tecnm.mx http://itochihuahua.mx](http://itochihuahua.mx)



Chihuahua, Chihuahua, septiembre de 2025

Dra. Rosaura Ruiz Gutiérrez

Titular de la SECIHTI

Por este conducto aprovecho la ocasión para saludarlo e informarle que a la fecha he obtenido el grado de **Maestría en Ciencias de Ingeniería Electrónica** en la División de Estudios de Posgrado e Investigación del Instituto Tecnológico de Chihuahua. Agradezco todo el apoyo brindado por SECIHTI ya que el otorgamiento de la beca para estudios de Posgrado, permitió dedicarme de tiempo completo al programa y con ello lograr el cumplimiento del objetivo principal del convenio establecido.

Sin otro particular por el momento, me es grato quedar de usted como su seguro servidor, no sin antes reiterar nuevamente todo mi agradecimiento.

A t e n t a m e n t e

Axel Fernando Tam Ruiz
Exbecario SECIHTI

AGRADECIMENTOS

En primer lugar, agradezco a mis padres Flor y Alejandro, a quienes debo mi persona, sin su amor y soporte incondicional este trabajo no hubiera podido ser realizado. Doy gracias a mi abuelita Mague por su calidez y cuidados, a mis mascotas Sheldon, Penny y Luna que con su compañía siempre me brindan alegría y el resto de mi familia y amigos, que me llenan de ánimo.

Gracias a mis compañeros de laboratorio Nirvana, Jose, Mike, Steve, Ariel, Kike, Eduardo, Javi, e Iván entre los que he encontrado amistad, aprendizaje y apoyo. Agradezco a mi director de tesis, el Dr. Mario Chacón, por ser maestro y guía en este trabajo, a los profesores que me enseñaron con dedicación a lo largo de mi formación y a los miembros del comité tutorial que aportaron al desarrollo de esta tesis.

Finalmente, quiero agradecer a todas las personas que directa o indirectamente han sido parte de este proceso y he omitido. Un río está formado por incontables gotas de agua cuyo paradero es imposible de determinar, sin embargo, su contribución es indispensable para hacer del agua un río.

RESUMEN

METODOLOGÍA PARA LA COMPRESIÓN DE REDES NEURONALES PROFUNDAS MEDIANTE TÉCNICAS DE INTELIGENCIA ARTIFICIAL EXPLICABLE

Ing. Axel Fernando Tam Ruiz
Maestría en Ciencias en Ingeniería Electrónica
División de Estudios de Posgrado e Investigación del
Instituto Tecnológico de Chihuahua
Chihuahua, Chih. 2025
Director de Tesis: Dr. Juan Alberto Ramírez Quintana

Las Redes Neuronales Profundas (*Deep Neural Networks*, DNNs) han revolucionado la inteligencia artificial en áreas como visión por computadora y reconocimiento de voz. No obstante, su elevado número de parámetros y capas interconectadas implican altos costos computacionales y energéticos, lo que dificulta su uso en dispositivos con recursos limitados. Para enfrentar este reto, se han propuesto métodos de compresión de modelos, como la poda estructural, cuyo propósito es reducir el tamaño y el costo computacional, manteniendo un desempeño competitivo.

Por otro lado, la Inteligencia Artificial Explicable (*eXplainable Artificial Intelligence*, XAI) proporciona interpretabilidad a modelos complejos, permitiendo identificar y analizar qué elementos de una red influyen de manera más significativa en sus predicciones. Por ello, integrar XAI en procesos de compresión aporta criterios más precisos para identificar parámetros redundantes y realizar podas informadas.

La presente investigación propone una metodología de poda estructural asistida por XAI, empleando principalmente los métodos EigenCAM y HiResCAM para estimar la importancia de filtros convolucionales. HiResCAM fue seleccionado por su capacidad de generar mapas de activación más precisos y estables, lo que permite identificar de manera confiable los filtros más relevantes en el modelo. Por su parte, EigenCAM ofrece explicaciones fácilmente interpretables por humanos, lo que aporta claridad en el análisis visual. Para validar la propuesta se utilizaron las arquitecturas representativas de visión por computadora ResNet-18, ResNet-50, VGG19-BN y DenseNet-121, entrenadas en dos bases de datos, CIFAR-10 y CIFAR-100.

Los resultados experimentales demostraron que la integración de técnicas de explicabilidad en el proceso de poda mejoraron la precisión con la que se identifican filtros prescindibles, logrando modelos más compactos y eficientes sin pérdidas significativas en exactitud. En algunos casos, tras el reentrenamiento, los modelos comprimidos incluso superaron la precisión original, lo que evidenció que la poda asistida por XAI no solo conserva el rendimiento, sino que puede favorecer la generalización. Además, se identificó que arquitecturas con conexiones residuales o densas, como ResNet y DenseNet, presentan mayor resiliencia a la poda.

CONTENIDO

LISTA DE FIGURAS	¡Error! Marcador no definido.
LISTA DE TABLAS	xvi
INTRODUCCIÓN	xviii
CAPÍTULO I. MARCO TEÓRICO	1
1.1 Redes neuronales profundas	1
1.1.1 Redes Neuronales Convolucionales (CNN).....	2
1.2 Compresión de modelos.....	5
1.2.1 Podado de modelo.....	6
1.2.2 Cuantificación	8
1.3 Inteligencia Artificial Explicable	11
1.3.1 Clasificación de técnicas de XAI.....	12
1.3.2 Métodos basados en backpropagation	14
CAPÍTULO II. ANTECEDENTES	15
2.1 Métodos tradicionales de compresión.....	15
2.2 Métodos de compresión utilizando XAI.....	19
2.3 Métodos de compresión en etapa de pre-entrenamiento.....	23
CAPÍTULO III. ENTORNO DE PRUEBAS	25
3.1 Descripción de arquitecturas seleccionadas.....	25
3.1.1 VGG19 con Batch Normalization.....	26
3.1.2 ResNet-18	28
3.1.3 ResNet-50	31
3.1.4 DenseNet-121	33
3.2 Conjuntos de datos utilizados	36
3.2.1 ImageNet-1K.....	36
3.2.2 CIFAR-10.....	38
3.2.3 CIFAR-100.....	42
3.3 Listado de modelos e hiperparámetros	43
3.4 Métricas de evaluación	45
3.4.1 Métricas de compresión	45
3.4.2 Métrica de desempeño	46

3.5	Frameworks y librerías	46
3.6	Protocolo experimental	47
CAPÍTULO IV. COMPARACIÓN DE MÉTODOS DE PODA DE REDES NEURONALES		
.....		50
4.1	Métodos de poda dependientes de los pesos.....	50
4.2	Métodos de poda basados en activaciones.....	51
4.3	Métodos de poda utilizando herramientas de optimización.....	52
4.4	Métodos de poda con regularización	53
4.5	Resultados experimentales y análisis comparativo.....	54
4.5.1	Análisis de la relación entre exactitud y parámetros podados	54
4.5.2	Impacto del re-entrenamiento en la exactitud vs reducción de FLOPs	57
4.5.3	Análisis de la eficiencia del re-entrenamiento tras poda	60
4.5.4	Modelos que mejoran el accuracy original tras poda y re-entrenamiento	63
4.6	Discusión general.....	64
CAPÍTULO V. EVALUACIÓN DE TÉCNICAS DE XAI PARA LA PODA ESTRUCTURAL..		67
5.1	Métodos basados en CAM.....	67
5.1.1	Class Activation Mapping.....	68
5.1.2	Gradient-weighted Class Activation Mapping.....	69
5.1.3	Variantes recientes de CAM.....	71
5.2	Evaluación comparativa de métodos basados en CAM.....	73
5.2.1	Evaluación cualitativa.....	73
5.2.2	Evaluación cuantitativa.....	77
5.2.3	Metodología para la evaluación comparativa de métodos XAI.....	83
5.3	Selección y descripción de métodos XAI	88
5.3.1	EigenCAM.....	92
5.3.2	High-Resolution Class Activation Mapping (HiResCAM)	94
CAPÍTULO VI. METODOLOGÍA PROPUESTA DE PODA ESTRUCTURAL BASADA EN XAI.....		96
6.1	Algoritmo general de poda estructural iterativa.....	96
6.2	Criterios de importancia basados en técnicas XAI	99
6.2.1	Importancia basada en EigenCAM	100

6.2.2	Importancia basada en HiResCAM	102
CAPÍTULO VII. RESULTADOS Y CONCLUSIONES.....		106
7.1	Resultados globales de la poda estructural	106
7.1.1	Métricas y umbrales de calidad.....	106
7.1.2	FLOPs vs accuracy	107
7.1.3	Parámetros vs accuracy.....	109
7.2	Comparativa y sensibilidad de los criterios de importancia	112
7.2.1	GradAbs vs EigenCAM.....	112
7.2.2	GradAbs vs métodos de poda de referencia.....	112
7.2.3	Impacto del número de mini-lotes en GradAbs	114
7.3	Efecto de la arquitectura y del dataset	117
7.3.1	Arquitecturas resilientes.....	117
7.3.2	Arquitecturas frágiles.....	118
7.3.3	Impacto del conjunto de datos.	119
7.4	Conclusiones.....	119
7.5	Trabajo futuro	121
REFERENCIAS		123
ANEXOS 129		
A	Resultados de selectividad.....	129
B	Resultados de continuidad.....	130
C	Resultados de enfoque.....	131
D	Resultados de LSE.....	131
GLOSARIO Y NOTACIÓN MATEMÁTICA		133

LISTA DE FIGURAS

CAPÍTULO I. MARCO TEÓRICO

Figura 1.1. Ejemplo de podado donde la unidad de podado es un filtro [12].....	8
Figura 1.2. (a) Podado de pesos. (b) Podado de neuronas [3].	8
Figura 1.3. Ejemplos de explicaciones para diferentes tipos de datos [21].	12

CAPÍTULO II. ANTECEDENTES

Figura 2.1. Flujo del marco de trabajo propuesto por Alqahtani et al. [31].....	20
Figura 2.2. Mapas de saliencia generados con AEM [33].	22

CAPÍTULO III. ENTORNO DE PRUEBAS

Figura 3.1. Comparación de campos receptivos de capas convolucionales con kernels de 3x3 (azul) y 5x5 (verde).....	27
Figura 3.2. Arquitectura de VGG19-BN.....	28
Figura 3.3. Agregación de capas que mapean funciones identidad.	29
Figura 3.4. Estructura de bloques residuales sin y con stride.	30
Figura 3.5. Arquitectura de ResNet-18.	31
Figura 3.6. Bloques residuales con cuello de botella.....	32
Figura 3.7. Arquitectura de ResNet-50.	33
Figura 3.8. Estructura de bloque denso.....	35
Figura 3.9. Estructura de capa de transición.	35
Figura 3.10. Arquitectura de DenseNet-121.	36
Figura 3.11. Ejemplos de imágenes pertenecientes a las clases <i>hamster</i> , <i>zebra</i> , <i>chain</i> y <i>corn</i> del conjunto de datos ImageNet-1K.	37
Figura 3.12. Imágenes pertenecientes a ImageNet-1K de las clases <i>hamster</i> , <i>zebra</i> , <i>chain</i> y <i>corn</i> preprocesadas.	38
Figura 3.13. Ejemplos de imágenes de las 10 clases de CIFAR-10 [45].	39
Figura 3.14. Arquitectura de VGG19-BN modificada para utilizarse con CIFAR-10.....	40
Figura 3.15. Arquitectura de ResNet-18 modificada para utilizarse con CIFAR-10.....	40
Figura 3.16. Arquitectura de ResNet-50 modificada para utilizarse con CIFAR-10.....	41
Figura 3.17. Arquitectura de DenseNet-121 modificada para utilizarse con CIFAR-10.....	41

CAPÍTULO IV. COMPARACIÓN DE MÉTODOS DE PODA DE REDES NEURONALES

Figura 4.1. Relación entre porcentaje de parámetros podados y precisión sin re-entrenamiento para VGG19-BN.	55
Figura 4.2. Relación entre porcentaje de parámetros podados y precisión sin re-entrenamiento para ResNet-18.....	55
Figura 4.3. Relación entre porcentaje de parámetros podados y precisión sin re-entrenamiento para ResNet-50.....	56
Figura 4.4. Relación entre porcentaje de parámetros podados y precisión sin re-entrenamiento para DenseNet-121.....	56
Figura 4.5. Relación entre reducción de FLOPs y precisión con re-entrenamiento para VGG19-BN.....	58
Figura 4.6. Relación entre reducción de FLOPs y precisión con re-entrenamiento para ResNet-18.	58
Figura 4.7. Relación entre reducción de FLOPs y precisión con re-entrenamiento para ResNet-50.	59
Figura 4.8. Relación entre reducción de FLOPs y precisión con re-entrenamiento para DenseNet-121.	59
Figura 4.9. Relación entre reducción de parámetros y número de épocas de re-entrenamiento para VGG19-BN.	61
Figura 4.10. Relación entre reducción de parámetros y número de épocas de re-entrenamiento para ResNet-18.....	61
Figura 4.11. Relación entre reducción de parámetros y número de épocas de re-entrenamiento para ResNet-50.....	62
Figura 4.12. Relación entre reducción de parámetros y número de épocas de re-entrenamiento para DenseNet-121.....	62

CAPÍTULO V. EVALUACIÓN DE TÉCNICAS DE XAI PARA LA PODA ESTRUCTURAL

Figura 5.1. Diagrama de obtención de CAM [60].	69
Figura 5.2. Diagrama de obtención de Grad-CAM [61].	70
Figura 5.3. Generación de CAMs de ResNet-50.	74

Figura 5.4. CAMs generados por distintos métodos a partir de las activaciones de salida de la etapa 1 de ResNet-50. (a) Imágenes originales, (b) GradCAM, (c) GradCAM++, (d) XGradCAM, (e) LayerCAM, (f) EigenCAM, (g) EigenGradCAM, (h) HiResCAM, (i) ScoreCAM, (j) AblationCAM, (k) FullGrad, (l) FEM, (m) KPCA-CAM, (n) FinerCAM. ...	75
Figura 5.5. CAMs generados por distintos métodos a partir de las activaciones de salida de la etapa 4 de ResNet-50. (a) Imágenes originales, (b) GradCAM, (c) GradCAM++, (d) XGradCAM, (e) LayerCAM, (f) EigenCAM, (g) EigenGradCAM, (h) HiResCAM, (i) ScoreCAM, (j) Abl.....	76
Figura 5.6. CAMs de las salidas de 4 etapas de ResNet-50 generados con EigenCAM.	77
Figura 5.7. Ejemplo visual de procedimiento para obtención de métrica de selectividad.....	78
Figura 5.8. Ejemplo visual del cálculo de la métrica de continuidad.	80
Figura 5.9. Ejemplo visual del cálculo de la métrica de enfoque.	81
Figura 5.10. Ejemplo visual del cálculo de la métrica LSE.....	83
Figura 5.11. Imágenes seleccionadas aleatoriamente del conjunto de datos ImageNet-1k con su respectiva clase.....	84
Figura 5.12. Ejemplos de CAMs generados a partir de mosaicos para calcular métrica de enfoque.....	85
Figura 5.13. Comparación de CAMs con bajas y altas activaciones.	87
Figura 5.14. CAMs de las salidas de 4 etapas de ResNet-50 generados con HiResCAM. ...	90
Figura 5.15. Comparación de Selectividad de EigenCAM y HiResCAM.....	91
Figura 5.16. Mapeo de características a un espacio discriminativo.....	92
Figura 5.17. Representación de cálculo de HiResCAM y GradCAM [62].	95

CAPÍTULO VII. RESULTADOS Y CONCLUSIONES

Figura 7.1. Relación entre reducción de FLOPs y precisión con re-entrenamiento para ResNet-50.	107
Figura 7.2. Relación entre reducción de FLOPs y precisión con re-entrenamiento para ResNet-18.	108
Figura 7.3. Relación entre reducción de FLOPs y precisión con re-entrenamiento para DenseNet-121.	108
Figura 7.4. Relación entre reducción de FLOPs y precisión con re-entrenamiento para VGG19-BN.....	109

Figura 7.5. Relación entre reducción de parámetros y precisión con re-entrenamiento para ResNet-50.	110
Figura 7.6. Relación entre reducción de parámetros y precisión con re-entrenamiento para DenseNet-121.	110
Figura 7.7. Relación entre reducción de parámetros y precisión con re-entrenamiento para ResNet-18.	111
Figura 7.8. Relación entre reducción de parámetros y precisión con re-entrenamiento para VGG19-BN.	111
Figura 7.9. Trade-off FLOPs vs accuracy en CIFAR-100 con variación de mini-lotes.	115
Figura 7.10. Trade-off FLOPs vs accuracy en CIFAR-10 con variación de mini-lotes.	115
Figura 7.11. Filtros podados por capa en ResNet-50 con factor de aceleración $1.5\times$ y variación de mini-lotes.	116
Figura 7.12. Filtros podados por capa en ResNet-50 con factor de aceleración $20\times$ y variación de mini-lotes.	116

LISTA DE TABLAS

CAPÍTULO I. ANTECEDENTES

Tabla 2.1. Resultados de los experimentos en ImageNet con ResNet-34/50/101 y MobileNet-V2 [30].	22
---	----

CAPÍTULO III. ENTORNO DE PRUEBAS

Tabla 3.1. Superclases y clases de CIFAR-100 [41].	43
Tabla 3.2. Hiperparámetros para el entrenamiento de las arquitecturas.	45
Tabla 3.3. Accuracy de los modelos pre-entrenados.	45

CAPÍTULO IV. COMPARACIÓN DE MÉTODOS DE PODA DE REDES NEURONALES

Tabla 4.1. Modelos entrenados que superan el <i>accuracy</i> original.	63
--	----

CAPÍTULO V. EVALUACIÓN DE TÉCNICAS DE XAI PARA LA PODA ESTRUCTURAL

Tabla 5.1. Resultados de la métrica de continuidad para cada método CAM evaluado en 10 imágenes (Etapa 4 de ResNet-50). Valores menores son mejores.	86
Tabla 5.2. Rankings de los métodos CAM según cada métrica.	86
Tabla 5.3. Métricas de evaluación de métodos de XAI y su ponderación.	88
Tabla 5.4. Ordenamiento de métodos de XAI con ponderación de rankings.	88

CAPÍTULO VII. RESULTADOS Y CONCLUSIONES

Tabla 7.1. Promedio de los máximos % de reducción de FLOPs de cada método logrando un $\Delta Acc \geq -1$.	113
Tabla 7.2. Promedio de los máximos % de reducción de parámetros de cada método logrando un $\Delta Acc \geq -1$.	113
Tabla 7.3. Casos que superan el <i>accuracy</i> original.	118
Tabla 7.4. Top-3 de mejores desempeños en ResNet-18 y VGG19-BN.	119

ANEXOS

Tabla A.1. Resultados de la métrica de selectividad para cada método CAM evaluado en 10 imágenes (Etapa 1 de ResNet-50). Valores menores son mejores.	113
Tabla A.2. Resultados de la métrica de selectividad para cada método CAM evaluado en 10 imágenes (Etapa 2 de ResNet-50). Valores menores son mejores.	11329

Tabla A.3. Resultados de la métrica de selectividad para cada método CAM evaluado en 10 imágenes (Etapa 3 de ResNet-50). Valores menores son mejores.	11829
Tabla A.4. Resultados de la métrica de selectividad para cada método CAM evaluado en 10 imágenes (Etapa 4 de ResNet-50). Valores menores son mejores.	119
Tabla B.1. Resultados de la métrica de continuidad para cada método CAM evaluado en 10 imágenes (Etapa 1 de ResNet-50). Valores menores son mejores.	119
Tabla B.2. Resultados de la métrica de continuidad para cada método CAM evaluado en 10 imágenes (Etapa 2 de ResNet-50). Valores menores son mejores.	119
Tabla B.3. Resultados de la métrica de continuidad para cada método CAM evaluado en 10 imágenes (Etapa 3 de ResNet-50). Valores menores son mejores.	119
Tabla C.1. Resultados de la métrica de enfoque para cada método CAM evaluado en 10 imágenes (Etapa 4 de ResNet-50). Valores mayores son mejores.	1191
Tabla D.1. Resultados de la métrica de LSE para cada método CAM evaluado en 10 imágenes (Etapa 1 de ResNet-50). Valores mayores son mejores.	1191
Tabla D.2. Resultados de la métrica de LSE para cada método CAM evaluado en 10 imágenes (Etapa 2 de ResNet-50). Valores mayores son mejores.	11932
Tabla D.3. Resultados de la métrica de LSE para cada método CAM evaluado en 10 imágenes (Etapa 3 de ResNet-50). Valores mayores son mejores.	119
Tabla D.4. Resultados de la métrica de LSE para cada método CAM evaluado en 10 imágenes (Etapa 4 de ResNet-50). Valores mayores son mejores.	119

INTRODUCCIÓN

En la última década, las Redes Neuronales Profundas (*Deep Neural Networks*, DNNs) se han consolidado como las principales aportadoras a los avances en inteligencia artificial (IA), logrando resultados sobresalientes en áreas como la visión por computadora, el procesamiento del lenguaje natural y el reconocimiento de voz [1]. Su capacidad para aprender representaciones complejas de los datos las ha posicionado como herramientas indispensables en la resolución de problemas del mundo real. No obstante, esta eficacia suele implicar un aumento considerable en el número de parámetros, la cantidad de operaciones requeridas y la profundidad de las arquitecturas, lo que deriva en altos requerimientos computacionales y energéticos, dificultando su implementación en dispositivos con recursos limitados y reduciendo su sostenibilidad [2].

La creciente demanda de aplicaciones basadas en IA en dispositivos móviles, sistemas embebidos y entornos con restricciones de hardware ha impulsado el interés por técnicas de compresión de modelos. Estas buscan reducir el tamaño, la latencia de inferencia y el consumo energético de las DNNs, manteniendo al mismo tiempo un rendimiento competitivo [3]. Entre las técnicas más relevantes destacan la poda de parámetros y la cuantificación, las cuales han demostrado ser efectivas para simplificar arquitecturas profundas sin comprometer de manera significativa su exactitud [4]. Sin embargo, un reto persistente en este campo consiste en determinar con precisión qué parámetros, filtros o estructuras de una red pueden eliminarse o transformarse sin deteriorar su capacidad predictiva.

En este contexto, ha emergido la Inteligencia Artificial Explicable (*eXplainable Artificial Intelligence*, XAI) como un área complementaria de gran relevancia. La XAI busca proporcionar interpretabilidad a los modelos de IA, permitiendo a los usuarios comprender los mecanismos internos detrás de las decisiones tomadas por las redes neuronales [5]. El uso de técnicas de explicabilidad no solo aporta transparencia y confianza en aplicaciones críticas, sino que también ofrece criterios informados para comprimir modelos. En particular, estas técnicas pueden emplearse como guías en procesos de compresión, identificando de manera más precisa las estructuras redundantes o irrelevantes dentro de una red [6].

La intersección entre compresión de modelos y XAI plantea un camino prometedor hacia el desarrollo de sistemas de aprendizaje profundo más eficientes, transparentes y adaptables.

Diversos estudios han explorado la integración de métodos explicativos como Layer-wise Relevance Propagation (LRP) o DeepLIFT en procesos de poda y cuantificación, demostrando que la incorporación de criterios interpretables puede mejorar la eficacia de la compresión y, en algunos casos, incluso mantener o superar el rendimiento del modelo original [6], [7]. Sin embargo, persisten desafíos relacionados con la generalización de estas metodologías a diferentes arquitecturas y conjuntos de datos, así como con la necesidad de garantizar un equilibrio adecuado entre interpretabilidad, eficiencia y exactitud.

En este marco, la presente tesis propone una metodología de poda estructural basada en técnicas de XAI, con el objetivo de reducir la complejidad de las DNNs sin comprometer de manera significativa su desempeño. Para ello, se integran métodos recientes de visualización e interpretabilidad, como EigenCAM y HiResCAM, seleccionados por su capacidad de generar mapas de activación más interpretables, precisos y estables, lo que permite estimar de forma confiable la relevancia de los filtros convolucionales y guiar así el proceso de poda estructurada. Se hipotetiza que el uso de visualizaciones derivadas de métodos de XAI durante los procesos de clasificación contribuirá a una selección más adecuada de las estructuras prescindibles, lo que resultará en modelos más compactos y eficientes.

El documento se organiza de la siguiente manera: en los Capítulos I y II se presentan el marco teórico y los antecedentes que fundamentan esta investigación, abordando a detalle las DNNs, las técnicas de compresión de modelos y los métodos de XAI. En el Capítulo III se describe el entorno experimental, incluyendo las arquitecturas seleccionadas, los conjuntos de datos utilizados y las métricas de evaluación. En el Capítulo IV se comparan distintos métodos tradicionales de poda, analizando su impacto en términos de rendimiento y eficiencia. El Capítulo V introduce y evalúa diversas técnicas de XAI aplicables a la poda estructural, mientras que en el Capítulo VI se presenta la metodología propuesta. Finalmente, en el Capítulo VII se discuten los resultados obtenidos, conclusiones y posibles líneas de trabajo futuro.

CAPÍTULO I. MARCO TEÓRICO

En este marco teórico se abordan tres temas fundamentales: Redes Neuronales Profundas (*Deep Neural Networks*, DNNs), compresión de modelos e Inteligencia Artificial Explicable (*eXplainable Artificial Intelligence*, XAI). La selección de estos temas se justifica por su relevancia y aplicabilidad directa en el objetivo principal de esta tesis.

Primero, se exploran las DNNs, debido a que constituyen la arquitectura sobre la cual se aplicarán las técnicas de compresión. En particular, investiga a las Redes Neuronales Convolucionales (*Convolutional Neural Networks*, CNNs), ya que son ampliamente utilizadas en tareas de visión por computadora. A continuación, se aborda el tema de la compresión de modelos, centrándose en dos técnicas principales: la poda y la cuantificación. Estas técnicas se han seleccionado debido a su potencial para reducir el tamaño y la complejidad computacional de las DNNs sin comprometer significativamente su rendimiento. Finalmente, se introduce al área de XAI, que busca proporcionar interpretabilidad a los modelos de IA, permitiendo a los usuarios entender y confiar en sus decisiones.

1.1 Redes neuronales profundas

Las DNNs son una subcategoría del aprendizaje profundo (*Deep Learning*, DL), que a su vez es una rama del aprendizaje automático (*Machine Learning*, ML). Inspiradas en el procesamiento de información del cerebro humano, las DNN se componen de múltiples capas de neuronas artificiales que permiten la interpretación jerárquica y compleja de los datos. Cada capa en una DNN aprende diferentes representaciones de los datos de entrada, lo que permite la automatización del aprendizaje de características, a diferencia de los métodos tradicionales de ML que requieren preprocesamiento y selección de características manuales [2].

Una DNN puede ser formalizada como una función $f: X \rightarrow Y$, donde X representa el espacio de entrada y Y el espacio de salida. La red se compone de múltiples capas $L^{(l)}$ que transforman una entrada x en una salida y mediante una serie de operaciones no lineales. Cada capa $L^{(l)}$ aplica una transformación $h_{(l)}$ a su entrada, y esta transformación generalmente incluye una función de activación σ y una operación lineal parametrizada por los pesos $W_{(l)}$.

$$h_{(l)} = \sigma(W_{(l)} \square h_{(l-1)} + b) \quad (1.1)$$

Donde $h_{(0)} = x$ es la entrada inicial y $h_{(n)} = y$ es la salida final de la red después de n capas. Las funciones de activación comunes incluyen ReLU, sigmoide y tanh, que permiten capturar patrones no lineales en los datos. A continuación, se presenta el tipo más común de DNN utilizada para tareas de visión por computadora, las redes neuronales convolucionales.

1.1.1 Redes Neuronales Convolucionales (CNN)

Las Redes Neuronales Convolucionales (CNN) están especialmente diseñadas para procesar datos con una estructura de cuadrícula, como las imágenes. Están diseñadas para automatizar y mejorar el proceso de extracción de características mediante el uso de operaciones de convolución, que permiten detectar patrones locales en los datos de entrada. Una CNN se compone típicamente de varias capas de diferentes tipos, incluyendo capas convolucionales, capas de *pooling* y capas completamente conectadas. A continuación, se describe la formalización matemática de cada tipo de capa [8].

1.1.1.1 Capa convolucional

Una capa convolucional aplica filtros a la entrada para extraer características locales. La entrada $T^{(l)}$ de la l -ésima capa convolucional se puede representar como un tensor de dimensiones $H_{in} \times W_{in} \times C_{in}$, donde H_{in} es la altura de la entrada, W_{in} es el ancho de la entrada y C_{in} el número de canales de la entrada.

De la misma manera, un filtro $F_k^{(l)}$ se ve como un tensor de 3 dimensiones $k_h \times k_w \times C_{in}$, donde k_h y k_w son las dimensiones del *kernel* $K^{(l)}$. El filtro extiende el *kernel* a todas las dimensiones de la entrada, es decir, se realiza la convolución sobre los C_{in} canales de la entrada.

Para controlar las dimensiones de la salida de la convolución se utilizan dos parámetros adicionales: el *stride* y el *padding*.

El *stride* determina el paso con el que se desplaza el filtro sobre la entrada. Un valor de *stride* mayor que 1 implica que el filtro se desplaza saltando posiciones, resultando en una reducción de

dimensiones de salida. Si el *stride* es 1, el filtro se desplaza de una posición a la siguiente sin dejar espacios entre ellas.

El *padding* es el proceso de agregar ceros alrededor de los bordes de la entrada. Si se realiza la convolución sin *padding*, las dimensiones de la entrada se reducen debido a que los filtros no se pueden aplicar completamente en los bordes. Aplicar *padding* a una entrada puede preservar o aumentar las dimensiones de la entrada al aplicarle convolución, dependiendo de los valores que se usen.

Matemáticamente, la salida $T^{(l+1)}$ en las coordenadas (i, j, k) para la operación de convolución para una entrada $T^{(l)}$ y el k -ésimo filtro $F_k^{(l)}$ se define como:

$$T^{(l+1)}(i, j, k) = \sigma \left(\sum_{c=1}^{C_{in}} \sum_{m=1}^{k_h} \sum_{n=1}^{k_w} T^{(l)}(iS + m, jS + n, c) F_k(m, n, c) + b_k \right) \quad (1.2)$$

Donde S es el *stride* de desplazamiento del *kernel* sobre la entrada.

Así bien, la salida de una capa convolucional $T^{(l+1)}$ tiene dimensiones $H_{out} \times W_{out} \times C_{out}$ donde H_{out} y W_{out} están dados por las ecuaciones (1.3) y (1.4) respectivamente y C_{out} se determina por el número de filtros aplicados. P es el *padding* aplicado a la entrada.

$$H_{out} = \left\lfloor \frac{H_{in} - k_h + 2P}{S} \right\rfloor + 1 \quad (1.3)$$

$$W_{out} = \left\lfloor \frac{W_{in} - k_w + 2P}{S} \right\rfloor + 1 \quad (1.4)$$

Un filtro puede interpretarse como un detector de características. Cuando una característica específica está presente en la entrada, el filtro produce valores altos en el mapa de características correspondiente y, por el contrario, si la característica no está presente, los valores en la salida son pequeños o nulos. Por esta razón, el resultado de aplicar un filtro se llama mapa de características. Una capa convolucional $L^{(l)}$ está caracterizada por los C_{out} filtros que la componen. Por cada filtro se genera un mapa de características $M_k^{(l)}$ que en conjunto forman la salida $T^{(l+1)}$.

1.1.1.2 Capa de Pooling

La capa de *pooling* reduce la dimensionalidad de la entrada al tomar el valor máximo (*max pooling*) o el valor medio (*average pooling*) de subáreas de la entrada que no se solapan entre sí. En CNNs, este tipo de capas recibe como entrada un mapa de características generado por una capa convolucional. Por medio de las capas de *pooling*, se reduce la dimensionalidad de estas características, lo cual disminuye el número de parámetros. Esto contribuye a reducir el riesgo de sobreajuste y permite extraer las características más sobresalientes obtenidas de la capa convolucional anterior.

Para *max pooling*, la operación se define como:

$$Y(i, j) = \max_{m, n} X(i + m, j + n) \quad (1.5)$$

donde:

- (i, j) son las coordenadas de la salida.
- (m, n) definen el tamaño de la ventana de *pooling*.
- X representa la entrada a la capa de *pooling*.
- Y representa la salida después de aplicar la operación de *max pooling*.

Por otro lado, para *average pooling* se utiliza el valor promedio de los elementos de la ventana de *pooling*, lo que implica la realización de más operaciones.

1.1.1.3 Capa de batch normalization

La capa de normalización por lotes (*Batch Normalization*) es una técnica que permite acelerar y estabilizar el entrenamiento de redes neuronales profundas mediante la normalización de las activaciones internas de cada capa. La idea detrás de esta técnica es reducir el problema de *internal covariate shift*, que ocurre cuando la distribución de las entradas de cada capa cambia constantemente durante el entrenamiento debido a las actualizaciones de parámetros en las capas anteriores. Esto puede provocar dificultades en la convergencia y la necesidad de utilizar tasas de aprendizaje muy pequeñas, lo que ralentiza el entrenamiento [9].

Formalmente, dado un mini-lote $B = \{x_1, \dots, x_m\}$, se calculan la media μ_B y la varianza σ_B^2 para cada característica de entrada, donde:

$$\mu_B = \frac{1}{m} \sum_{i=1}^m x_i \quad (1.6)$$

$$\sigma_B^2 = \frac{1}{m} \sum_{i=1}^m (x_i - \mu_B)^2 \quad (1.7)$$

Luego, cada elemento x_i del lote se normaliza:

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \delta}} \quad (1.8)$$

donde δ es un pequeño valor añadido para evitar la división entre cero. Finalmente, se introducen parámetros de escalado γ y desplazamiento β , aprendidos durante el entrenamiento, que permiten restaurar o ajustar la distribución de los datos si es necesario:

$$y_i = \gamma \hat{x}_i + \beta \quad (1.9)$$

1.1.1.4 Capa completamente conectada

Las capas completamente conectadas conectan cada neurona de la capa anterior con cada neurona de la capa actual, similar a las redes neuronales tradicionales. Estas capas permiten que las redes convolucionales combinen las características extraídas por capas anteriores para tomar decisiones finales, tales como clasificaciones o predicciones. La salida de una capa completamente conectada se define como en (1.1).

Estas capas completamente conectadas generalmente son usadas hacia el final de la arquitectura de una CNN, especialmente cuando la red está diseñada para tareas de clasificación. Sin embargo, también pueden contribuir significativamente al número total de parámetros del modelo, lo que aumenta la complejidad computacional y la memoria requerida.

1.2 Compresión de modelos

La compresión de modelos (*Model Compression*) hace referencia al conjunto de técnicas y métodos utilizados dentro del DL que buscan reducir el tamaño, la complejidad computacional y el consumo energético de modelos de DNNs sin comprometer significativamente su rendimiento [4]. La compresión de modelos de DNN se ha convertido en un campo de estudio necesario para implementar estas redes en dispositivos con recursos limitados debido a la creciente demanda de aplicaciones de inteligencia artificial en teléfonos móviles y otros sistemas embebidos. A continuación, se explican la poda y la cuantificación de parámetros, dos de las técnicas mayormente utilizadas.

1.2.1 Podado de modelo

El podado de modelos (*Pruning*) es una de las técnicas principales para reducir la complejidad computacional y los requisitos de almacenamiento de las DNN, particularmente en aplicaciones de visión por computadora y procesamiento del lenguaje natural. El objetivo del podado es eliminar unidades del modelo menos significativas sin una degradación significativa del rendimiento del modelo. A la acción de establecer a cero ciertos pesos o eliminar físicamente conexiones de la red se le conoce como acción de dispersión o acción de esparsificación (*sparsifying action*). Este proceso no solo mejora la eficiencia de inferencia, sino que también puede contribuir a una mejor generalización al reducir el sobreajuste.

En los métodos de podado, se identifican cinco elementos fundamentales [10]:

1. Modelo base: Este aspecto se refiere a si la poda se realiza en un modelo pre-entrenado o en un modelo inicializado aleatoriamente.
2. Granularidad de la dispersión (*Sparsity granularity*): Este aspecto define el tamaño del grupo de pesos más pequeño que se podará. La granularidad puede variar desde la eliminación de pesos individuales hasta la eliminación de neuronas completas, filtros o incluso capas enteras en la red.
3. Razón de poda (*Pruning ratio*): Indica la cantidad o el porcentaje de pesos que se eliminarán del modelo.
4. Criterio de poda (*Pruning criterion*): Define la medida o criterio utilizado para seleccionar los grupos de poda que se eliminarán del modelo.

5. Programa de poda (*Pruning schedule*): Describe cómo se planifica y aplica la dispersión a lo largo del proceso de poda. Esto incluye la secuencia y frecuencia con la que se realiza la poda durante el entrenamiento del modelo.

Existen diferentes formas de clasificar un método de poda en redes neuronales según sus elementos fundamentales. En primera instancia, se pueden clasificar según el momento de poda, que puede ser después del entrenamiento del modelo base o durante la inicialización del modelo. La poda después del entrenamiento ha sido la norma durante las últimas décadas, mientras que la poda en la inicialización ha ganado atención recientemente por su potencial de igualar el rendimiento de los modelos densos.

Una segunda forma de clasificación de algoritmos de poda es según su criterio de poda, que puede basarse en la importancia de los pesos o en la regularización. La poda basada en importancia selecciona pesos para eliminar según criterios predefinidos como la magnitud de los pesos. La poda basada en regularización añade términos de penalización en la función objetivo para llevar los pesos no importantes a cero antes de eliminarlos.

Por último, la clasificación más comúnmente utilizada es poda estructurada y la poda no estructurada, que se basa en la granularidad de la dispersión utilizada. El podado estructurado se refiere a la eliminación de unidades completas como canales, capas, bloques o filtros. Este método puede simplificar de forma significativa la arquitectura de la red sin requerir software o hardware especializado para manejar matrices dispersas, resultando en reducciones importantes en el uso de memoria y en mejoras en la velocidad de inferencia. Al realizar el podado estructurado, se busca preservar la integridad estructural de la red [11]. En la Figura 1.1 se muestra un ejemplo de podado de filtros en un modelo de CNN, que resulta en un modelo más compacto.

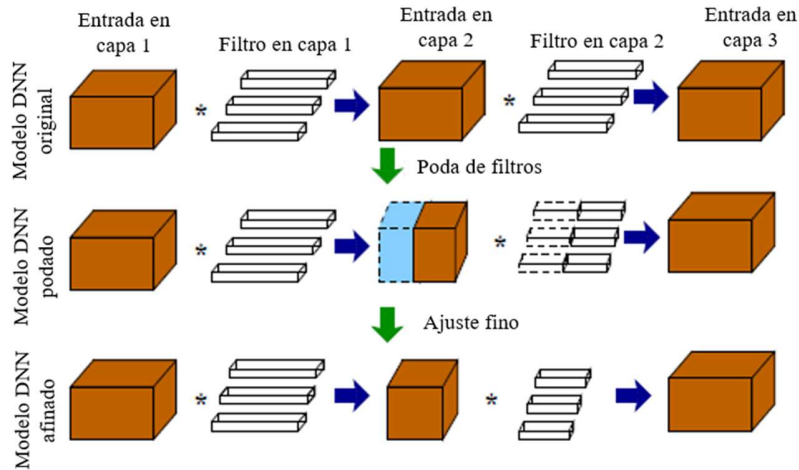


Figura 1.1. Ejemplo de podado donde la unidad de podado es un filtro [12].

Por otro lado, el podado no estructurado se centra en eliminar pesos o neuronas individuales a lo largo de la red sin prestar atención a la estructura de la misma (Figura 1.2). Esto resulta en matrices de pesos dispersas que, si bien pueden no aprovechar eficientemente las optimizaciones de hardware convencionales, pueden proporcionar una reducción significativa en el tamaño del modelo con un impacto mínimo en su rendimiento. El desafío principal de esta técnica radica en la gestión de la dispersión resultante para mantener o mejorar la eficiencia computacional, además de que, mientras que el podado no estructurado requiere de un ajuste fino para recuperar la exactitud, el podado estructurado permite un entrenamiento completo desde el principio [4].

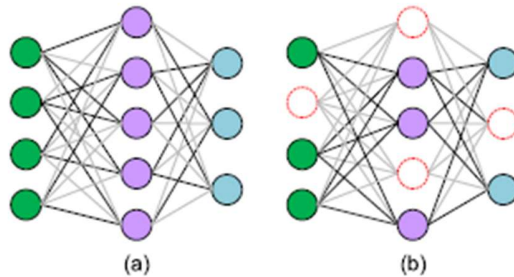


Figura 1.2. (a) Podado de pesos. (b) Podado de neuronas [3].

1.2.2 Cuantificación

En el área de procesamiento de señales, la cuantificación es un proceso que convierte valores de un espacio continuo en un conjunto de valores discretos. Por otra parte, en el campo de la compresión de modelos, la cuantificación hace uso de este principio para transformar los componentes numéricos de un modelo de DNN, generalmente pesos y/o activaciones, de una alta

precisión, como flotantes de 32 bits, a formatos de menor precisión, como enteros de 8 bits, sin causar un impacto significativo en el desempeño del modelo. Esta reducción en la precisión de los datos no solo disminuye el tamaño del modelo y los requisitos de ancho de banda, sino que también puede acelerar la inferencia y reducir el consumo de energía, especialmente en dispositivos con capacidad de cálculo limitada [13].

Para caracterizar los diferentes métodos de cuantificación se han identificado seis elementos fundamentales que se listan a continuación [14].

1. Niveles de cuantificación: Estos son los valores discretos a los que se mapean los valores continuos.

La cuantificación uniforme utiliza un mapeo equidistante de valores continuos a niveles discretos. En contraste, la cuantificación no uniforme asigna valores a niveles discretos no equidistantes, permitiendo una representación más precisa de datos distribuidos de manera desigual.

2. Rango: El rango se refiere al intervalo de valores continuos que se cuantificarán. Es crucial definir este rango adecuadamente para asegurarse de que los valores más significativos del modelo estén representados dentro de los niveles de cuantificación.

En la cuantificación simétrica, los valores se mapean dentro de un rango simétrico respecto al origen. La cuantificación asimétrica utiliza un rango no simétrico, que puede ser más eficiente cuando los datos están sesgados, como activaciones de ReLU.

3. Función de cuantificación: Esta función define cómo se realiza el mapeo de los valores continuos de precisión completa a los niveles discretos de cuantificación.
4. Algoritmos de calibración: Los algoritmos de calibración se utilizan para determinar los rangos óptimos y los niveles de cuantificación para los pesos y activaciones del modelo. Estos algoritmos pueden usar estadísticas recopiladas durante el entrenamiento o en un conjunto de datos de calibración.

La cuantificación estática calcula los rangos de cuantificación antes de la inferencia usando un conjunto de datos de calibración, mientras que la cuantificación dinámica ajusta los rangos en tiempo real durante la inferencia, lo que puede mejorar la precisión a costa de mayor complejidad computacional.

5. Granularidad de la cuantificación: Este aspecto define el nivel de detalle al que se aplica la cuantificación en el modelo de DNN. Puede realizarse a diferentes niveles de granularidad, incluyendo por capa (*layerwise*), por grupo (*groupwise*), por canal (*channelwise*), y por sub-canal (*sub-channelwise*). Cada nivel de granularidad ofrece un *trade-off* diferente entre precisión y sobrecarga computacional.
6. Programa de cuantificación: Describe cómo y cuándo se aplica la cuantificación durante el proceso de entrenamiento del modelo.

Esto puede incluir la cuantificación post-entrenamiento (*Post-Training Quantization*, PTQ), donde el modelo se cuantifica después de haber sido entrenado, o la cuantificación consciente del entrenamiento (*Quantization-Aware Training*, QAT), donde el modelo se entrena teniendo en cuenta los efectos de la cuantificación, por lo que después de cada actualización de gradiente, los parámetros son cuantificados nuevamente [15].

Una de las funciones de cuantificación más populares se muestra en la ecuación (1.10).

$$Q(r) = \text{Int}\left(\frac{r}{S}\right) - Z \quad (1.10)$$

Donde r es el valor de entrada en precisión completa que se desea cuantificar, S es el factor de escala, $\text{Int}()$ es una función que mapea un valor real a un valor entero mediante una operación de redondeo y Z es el punto cero entero, entero que centra la cuantificación en torno a un valor específico, facilitando la reconstrucción del valor original.

Este método de cuantificación se considera uniforme debido a que el factor de escala S divide todo el rango de los valores de entrada r en un número constante de particiones equidistantes y se calcula de la siguiente manera.

$$S = \frac{\zeta - \alpha}{2^b - 1} \quad (1.11)$$

Donde $[\alpha, \zeta]$ son los límites inferior y superior del rango de cuantificación, respectivamente, b es el ancho de bits de la cuantificación, por lo que $2^b - 1$ determina cuántas particiones se crean dentro del rango $[\alpha, \zeta]$. Si $-\alpha \neq \zeta$, la cuantificación caería dentro del esquema de cuantificación asimétrica, mientras que si $-\alpha = \zeta$, se consideraría cuantificación simétrica. El rango de cuantificación $[\alpha, \zeta]$ se determina por medio de los algoritmos de calibración, que van desde simplemente tomar el valor máximo absoluto de r , hasta el uso de métodos más sofisticados como la divergencia de Kullback-Leibler (KL) para minimizar la pérdida de información entre los valores originales de punto flotante y los valores que pueden ser representados por el formato cuantificado, o establecer el rango a un percentil de la distribución de valores absolutos vistos durante la calibración [16].

Por otra parte, se han explorado otras formas de cuantificación a través del *Clustering* o *Parameter Sharing*, a partir de algoritmos de agrupamiento como *k-means*, se cuantifican los parámetros y posteriormente pueden ser descomprimidos utilizando ya sea una tabla de búsqueda o una transformación lineal [17]. Otro de los enfoques que se han tomado, ha sido el representar los parámetros de mayor relevancia con mayor precisión, mientras que aquellos parámetros irrelevantes son guardados con menor precisión, conocido como *Mixed-Precision Quantization* [18].

1.3 Inteligencia Artificial Explicable

La Inteligencia Artificial Explicable se refiere al conjunto de procesos y métodos que da interpretabilidad humana a procesos o comportamientos internos de modelos de Inteligencia Artificial (IA). Esto permite a los usuarios humanos comprender y confiar en los resultados y salidas producidos por algoritmos de aprendizaje automático. La XAI busca resolver el problema de falta de transparencia en modelos de IA, común en soluciones de aprendizaje profundo, que, en aplicaciones como en sistemas de asistencia médica o manejo de autos autónomos, se convierte un problema crítico [5].

Las técnicas de XAI ofrecen explicaciones en conjunto a la salida de un modelo de IA. Las explicaciones se pueden entender como información útil para un público específico, y esta información se relaciona con las decisiones tomadas por un modelo de IA. Estas explicaciones deben ayudar al usuario a satisfacer una necesidad particular, como mejorar la arquitectura del modelo o justificar las decisiones tomadas por el mismo [19]. Así, para que una explicación sea considerada práctica, debe proporcionar uno de los siguientes tres aspectos, (1) una comprensión de los mecanismos y predicciones del modelo, (2) una visualización de las reglas de discriminación del modelo, o (3) indicaciones sobre qué factores podrían alterar el funcionamiento del modelo [20]. En la Figura 1.3 se dan ejemplos de diferentes explicaciones según el tipo de dato y método utilizado.

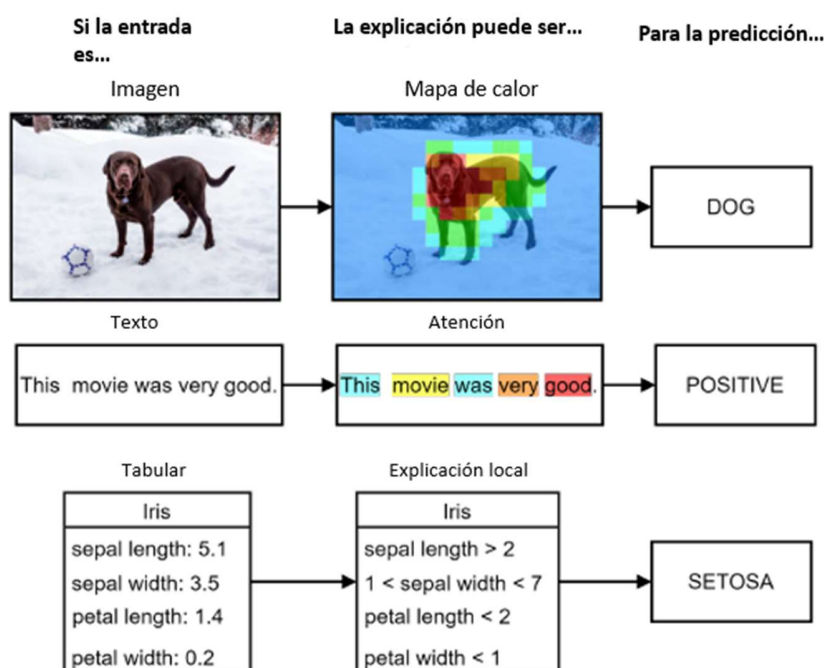


Figura 1.3. Ejemplos de explicaciones para diferentes tipos de datos [21].

1.3.1 Clasificación de técnicas de XAI

Las técnicas de XAI se pueden clasificar según diferentes criterios fundamentales. De acuerdo con una revisión de la literatura [22], las técnicas de XAI se pueden organizar en tres categorías principales: alcance, metodología y uso.

Primero, el alcance de las técnicas de XAI se puede dividir en dos subcategorías: local y global. Las técnicas locales se enfocan en la explicación de instancias individuales de datos, generando un mapa de explicación por cada instancia de datos δ . Ejemplos comunes incluyen LIME (*Local Interpretable Model-agnostic Explanations*) y SHAP (*SHapley Additive exPlanations*), que proporcionan explicaciones detalladas para predicciones específicas en lugar de ofrecer una visión general del modelo. Por otro lado, las técnicas globales intentan entender el modelo en su totalidad. Generalmente, toman un grupo de instancias de datos para generar uno o más mapas de explicación que proporcionen una visión global del comportamiento del modelo. Métodos como los árboles de decisión globales o las redes de reglas se utilizan para ofrecer una perspectiva completa.

En segundo lugar, la metodología utilizada en las técnicas de XAI se puede clasificar en métodos basados en *backpropagation* y métodos basados en perturbación. Los métodos basados en *backpropagation* se centran en la lógica algorítmica que depende de los gradientes que se retropropagan desde la capa de predicción de salida hasta la capa de entrada, en lugar de propagar errores, estos métodos propagan la importancia relevante a través de las capas de la red para determinar cómo cada característica de entrada contribuye a la salida final. Este enfoque incluye métodos como Grad-CAM (*Gradient-weighted Class Activation Mapping*), *Guided Backpropagation* y *Layer-wise Relevance Propagation* (LRP). Los métodos basados en perturbación, en cambio, dependen de cambios aleatorios o cuidadosamente elegidos en las características de la instancia de datos de entrada. Ejemplos de este enfoque incluyen LIME, donde se generan perturbaciones para evaluar la importancia de cada característica al observar cómo cambian las predicciones. Este enfoque se basa en la premisa de que cambiar partes importantes de la entrada debería tener un impacto significativo en la salida [21].

Por último, el uso de las técnicas de XAI puede ser intrínseco o *post-hoc*. En el uso intrínseco, la explicabilidad está integrada en la propia arquitectura de los modelos de IA y generalmente no es transferible a otras arquitecturas. Los modelos como los árboles de decisión y las regresiones lineales son intrínsecamente explicables debido a su naturaleza. Por otro lado, en el uso *post-hoc*, el algoritmo XAI no depende de la arquitectura del modelo y se puede aplicar a redes neuronales ya entrenadas. Este tipo de técnicas pueden aplicarse a cualquier modelo para proporcionar explicaciones después de que el modelo ha sido entrenado.

1.3.2 Métodos basados en *backpropagation*

Los métodos basados en *backpropagation* son especialmente relevantes para la compresión de modelos debido a su capacidad para ofrecer explicaciones de cómo los modelos complejos, como las redes neuronales profundas, toman decisiones. Estos enfoques permiten visualizar la contribución de cada característica de entrada a la predicción final mediante el uso de gradientes, lo que puede ayudar a identificar características redundantes o menos importantes, cruciales para la compresión de modelos. Además, los métodos basados en gradientes son útiles para la compresión porque la mayoría se centran en la visualización de activaciones de neuronas individuales con alta influencia en la salida del modelo [22]. Por ejemplo, LRP asigna una relevancia a cada neurona en las capas intermedias, mostrando cómo cada neurona contribuye a la salida final. Esta técnica ha sido utilizada como criterio de poda, ya que indica la importancia de las neuronas, permitiendo identificar cuáles se pueden eliminar sin afectar significativamente el rendimiento del modelo [6].

CAPÍTULO II. ANTECEDENTES

En la era actual, caracterizada por un crecimiento exponencial en la capacidad de procesamiento y la generación de datos, se ha vuelto imprescindible desarrollar modelos que no solo sean precisos sino también eficientes y sostenibles. La eficacia de las DNNs en el manejo de problemas reales se ve contrarrestada por su considerable tamaño y demanda de recursos, lo que impulsa la necesidad de técnicas de compresión que mitiguen estos inconvenientes sin deteriorar el rendimiento.

En esta sección, se inicia con una visión general de las técnicas y desafíos asociados a la compresión de DNN, destacando la importancia de reducir su tamaño y complejidad sin comprometer el rendimiento. Se sigue con una revisión de los métodos tradicionales de compresión, como la poda y la cuantificación, que han sentado las bases para aproximaciones más innovadoras. Posteriormente, se explora cómo la Inteligencia Artificial Explicable (eXplainable Artificial Intelligence, XAI) se integra en estos procesos para mejorar la eficacia de la compresión. Finalmente, se muestran estrategias específicas aplicadas durante la etapa de pre-entrenamiento, contrastando con los acercamientos anteriores.

2.1 Métodos tradicionales de compresión

Dentro de los métodos tradicionales de compresión de redes neuronales se encuentra el podado de redes. Esta técnica fue explorada inicialmente por LeCun et al. [23] con el propósito de adaptar el tamaño de una red neuronal eliminando pesos no esenciales, mejorando así la generalización, reduciendo los ejemplos de entrenamiento necesarios y acelerando el proceso de aprendizaje y clasificación. La metodología utilizada, denominada *Optimal Brain Damage* (OBD), se basaba en la eliminación selectiva de pesos utilizando información de las segundas derivadas del error de entrenamiento para evaluar la relevancia de los pesos. Este enfoque permitió una reducción significativa en la complejidad de la red sin comprometer, y en algunos casos mejorando, su rendimiento.

Otro trabajo pionero en el campo del podado de redes fue realizado por Hassibi et al. [24], quienes desarrollaron el *Optimal Brain Surgeon* (OBS), una técnica que mejora notablemente respecto a OBD y métodos basados en la magnitud de los pesos. OBS utilizó información de segundo orden del error para realizar una poda más precisa de la red, aprovechando la matriz

Hessiana, que corresponde a la matriz de segundas derivadas de la función de error respecto a los pesos y describe cómo cambia la curvatura de la superficie de pérdida. Al no imponer suposiciones restrictivas sobre la forma de esta matriz, OBS pudo identificar con gran precisión qué pesos podían eliminarse minimizando el incremento del error de entrenamiento. Como resultado, OBS logró reducciones de peso del 90%, 76% y 62% en tres problemas de referencia sin necesidad de entrenamiento adicional después de cada poda, demostrando una generalización superior en los datos de prueba.

Desde la introducción del podado por LeCun et al. en 1990 [23], el interés por el podado de redes ha continuado creciendo, abordándose desde múltiples enfoques que han refinado y ampliado las técnicas disponibles. Según la revisión realizada por He y Xiao [25], las técnicas de podado estructurado pueden clasificarse en varias categorías que se enfocan en diferentes aspectos de los modelos de redes neuronales. Entre los métodos utilizados se encuentran los dependientes de los pesos, que evalúan la importancia de los filtros de la red neuronal a partir de sus valores de peso. Estos métodos se diferencian por cómo abordan la medición de importancia de los filtros en las redes, algunos utilizando la magnitud de los pesos (norma de filtros), mientras que otros examinan las relaciones entre múltiples filtros (correlación de filtros) para identificar redundancias. Por ejemplo, en el método de Podado de Filtros a través de la Mediana Geométrica (FPGM) [26], se identifica que los filtros cercanos a la mediana geométrica son redundantes porque representan información común compartida por todos los filtros en la misma capa, lo que permite su eliminación sin una influencia significativa en el rendimiento del modelo. Otros métodos, como la Compresión Estructurada basada en Datos (RED) [27], utilizan técnicas sin necesidad de datos de entrada para fusionar filtros redundantes.

Por otro lado, los métodos de podado basados en activación se centran en utilizar los mapas de activación para tomar decisiones de podado. Estos métodos pueden actuar en diferentes niveles de detalle, como la poda a nivel de la capa actual (por ejemplo, método CP) o a nivel de capas adyacentes (por ejemplo, ThiNet y AOFD). Además, existen enfoques que evalúan el impacto global de la eliminación de un filtro en todas las capas, como NISP, que minimiza el error de reconstrucción en la Capa de Respuesta Final. Estos métodos aprovechan las características específicas de la activación de las neuronas para identificar la redundancia y eliminar los canales menos importantes sin sacrificar el rendimiento del modelo [25].

Además, la categoría de Herramientas de Optimización integra herramientas para encontrar o inducir dispersión estructurada durante el proceso de podado. Por ejemplo, se utilizan técnicas como la Expansión Taylor, que aproxima el cambio en la pérdida al podar estructuras como filtros o canales, y el Enfoque Bayesiano Variacional, que determina la importancia del filtro explotando las distribuciones a priori y posterior. También se emplean métodos como *Stochastic Gradient Descent* (SGD) y *Alternating Direction Method of Multipliers* (ADMM), que modifican la regla de actualización del gradiente para detectar filtros redundantes, proporcionando una variedad de herramientas para optimizar el proceso de podado.

Por otra parte, el podado dinámico, se puede llevar a cabo tanto durante el entrenamiento como durante la inferencia, adaptándose a diferentes escenarios y requisitos de aplicación. Métodos como SFP y GDP-Lin mantienen una máscara de podado dinámica durante el entrenamiento, mientras que RNP y FBS realizan podado dinámico durante la inferencia, ajustándose a diferentes muestras de entrada en tiempo real [25].

Otra de las técnicas comunes que impactan en la compresión de un modelo es la cuantificación. Gholami et al. [14] proporcionan una revisión exhaustiva de este campo en su trabajo, donde examinan métodos para minimizar el número de bits requeridos y maximizar la precisión de los cálculos asociados a las redes neuronales. El artículo presenta un abanico de estrategias de cuantificación, destacando la cuantificación uniforme y no uniforme, la cuantificación simétrica y asimétrica, y algoritmos de calibración de rango que pueden ser estáticos o dinámicos.

La granularidad de la cuantificación es también de gran importancia, siendo la cuantificación por capas, grupos o canales. Entre las técnicas más refinadas se encuentran la cuantificación no uniforme y métodos de ajuste fino post-cuantificación, incluyendo entrenamiento consciente de la cuantificación, cuantificación post-entrenamiento y cuantificación sin disparo, esta última de particular relevancia cuando los datos son extensos o sensibles. Mientras que la cuantificación estocástica se aborda como una alternativa a la cuantificación determinista durante la inferencia, proporcionando a las DNN la capacidad de explorar más durante el entrenamiento.

La revisión hecha por Ghimire [28] sobre CNNs destaca la descomposición tensorial como estrategia clave para compactar modelos sin sacrificar rendimiento. Distingue dos enfoques

principales: la descomposición matricial de bajo rango y la tensorizada. La primera, con métodos como la Descomposición en Valores Singulares (SVD), permite simplificar matrices de pesos en productos de matrices más pequeñas. Las técnicas innovadoras en este ámbito incluyen entrenamientos especializados que integran aproximaciones de bajo rango y esquemas conjuntos de descomposición, evitando el uso repetitivo de SVD y potenciando la compresión de capas similares en conjunto.

La descomposición tensorizada extiende estas técnicas a estructuras multidimensionales, con el método de Tucker reduciendo los tensores a componentes más manejables y la descomposición CP desglosándolos en sumas de tensores de rango-1. Los enfoques de *Tensor Train* (TT) y *Tensor Ring* (TR) son notables por su manejo eficaz de tensores de orden superior, adecuados especialmente para redes neuronales recurrentes (RNN). Aunque la descomposición TT puede mermar la precisión en CNNs, la selección cuidadosa de rangos puede contrarrestar este efecto, facilitando la compresión de modelos tridimensionales.

Tomando en cuenta el amplio catálogo de técnicas populares utilizadas para comprimir modelos de DNN, Marinó et al. [29] han evaluado comparativamente el desempeño de tres técnicas de compresión: poda de conexiones, cuantificación mediante compartición de pesos en cuatro variantes, y factorización de rango bajo, aplicadas a dos CNNs preentrenadas, VGG19 y DeepDTA, en cinco bases de datos distintas abarcando clasificación y regresión. La metodología empleada consistió en aplicar individual y combinadamente dichas técnicas, evaluando su impacto en términos de exactitud (*accuracy*), error cuadrático medio (MSE) y tasa de ocupación, considerando distintos tipos de capas. Los resultados obtenidos revelaron que la cuantificación global, en comparación con la individual por capas, generalmente produce una mayor tasa de compresión y, en la mayoría de los casos, mantiene o mejora la precisión. Además, se identificó que la poda de pesos, combinada con cuantificación, resulta particularmente efectiva en capas completamente conectadas (FC), mientras que la cuantificación en capas convolucionales afecta el desempeño. Por otro lado, la factorización de rango bajo emerge como una alternativa válida y más rápida en problemas de clasificación. Estas estrategias permitieron reducir considerablemente la ocupación de los modelos, en algunos casos hasta aproximadamente 20 veces, sin comprometer o incluso mejorando el rendimiento de referencia.

2.2 Métodos de compresión utilizando XAI

La compresión de modelos utilizando XAI ha emergido como un enfoque prometedor para reducir la complejidad de las DNN. Al emplear métodos de XAI, se busca no solo comprimir el modelo, sino también comprender la relevancia de las neuronas y unidades individuales en el proceso de toma de decisiones. En esta sección, se exploran los distintos enfoques que utilizan XAI para la compresión de modelos de DNN.

Por ejemplo, en trabajos recientes se diseñaron dos algoritmos utilizando DeepLIFT, para determinar la relevancia de las neuronas en la poda de una DNN [7], además se propuso la métrica *DeepLIFT Weighted Mean Squared Error* (DWMSE), para el algoritmo de agrupamiento *k-means* utilizado para la cuantificación. El proceso incluyó una estrategia de poda local basada en DeepLIFT y una evaluación de sensibilidad para cada capa, culminando en una poda global optimizada según la sensibilidad y la importancia calculada con DeepLIFT. Esta técnica, aplicada al modelo AlexNet, utilizado con la base de datos Flowers 102, permitió lograr una reducción de la complejidad del modelo: específicamente, los parámetros se redujeron en 30 veces a un total de 2.03 millones, y los GFLOPs se redujeron a 0.47, tres veces menores comparados con el modelo no podado, manteniendo una precisión de 0.698. Por otro lado, en el estudio realizado por Yeom et al. [6], se desarrolló un método de poda para DNN fundamentado en la técnica de LRP. Esta metodología se destaca por asignar valores de relevancia a las unidades individuales de la red, indicando su impacto en las decisiones finales del modelo y se propone como un nuevo criterio para el podado. El análisis comparativo del nuevo enfoque frente a métodos tradicionales se realizó bajo dos circunstancias diferentes: con y sin re-entrenamiento posterior a la poda. Los experimentos, aplicados a seis datasets y cuatro estructuras de CNN (VGG-16, AlexNet, ResNet-18 y ResNet-50), revelaron que LRP ofrece una poda más eficaz en contextos donde el ajuste fino no es viable.

En una investigación adicional realizada por Soroush et al. [30] se señala que determinar la importancia de unidades de un modelo con DeepLIFT, como se hace en el trabajo realizado por Sabih et al. [7], implica un conocimiento especializado para seleccionar una referencia de activación utilizada por DeepLIFT, que puede variar dependiendo del conjunto de datos. En cambio, optan por la técnica LRP para una evaluación más consistente y directa de la importancia

de los parámetros en la poda y cuantificación de modelos. Los experimentos con una DNN de tres capas densas y activadas con ReLU lograron reducir el modelo de 8 MB a 0.82 MB, obteniendo un *accuracy* del 90.125%. Se comparó la implementación del algoritmo desarrollado utilizando LRP [6] con el propio, logrando una mayor precisión (90.125% vs. 82.01%) al comprimirlo a la misma tasa.

Por su parte Alqahtani et al. [31] presentaron un marco para la compresión y aceleración de CNN al podar unidades ocultas individuales. A diferencia de los métodos convencionales, este enfoque emplea la activación de mapas de características para detectar información valiosa y partes semánticas esenciales para evaluar la importancia de los mapas de características, inspirado en la interpretación de redes neuronales. Se propuso una técnica de votación mayoritaria basada en el grado de alineación entre un concepto semántico y las representaciones de unidades ocultas individuales para evaluar cuantitativamente la importancia de los mapas de características. Además, se sugirió un método para estimar nuevos núcleos de convolución basados en los canales cruciales restantes, logrando así una compresión efectiva de CNN, el flujo de la metodología se observa en la Figura 2.1.

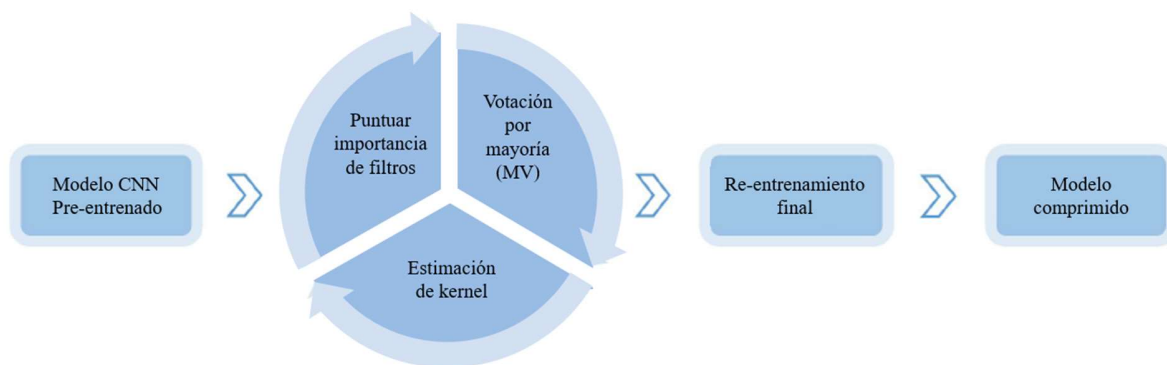


Figura 2.1. Flujo del marco de trabajo propuesto por Alqahtani et al. [31].

Los resultados experimentales destacan la eficacia de los criterios de selección de filtros propuestos, superando a los métodos de referencia del estado del arte. Se demostró una reducción del 50% en FLOPs en CIFAR-10 con solo un 0.86% de caída en precisión en el modelo VGG-16. Para ResNet, se logró una reducción del 30% en FLOPs con descensos mínimos en precisión.

En el estudio de Yao et al. [32], se introduce un enfoque para la reducción efectiva de DNNs, poniendo énfasis en la claridad interpretativa de los modelos. A través del uso de la técnica de maximización de activación, lograron aislar y examinar individualmente cada filtro de las DNNs, permitiendo una comprensión detallada de su función y relaciones. Este análisis detallado guio posteriormente una metodología sistemática y basada en el conocimiento para la poda de la red, diferenciándose de los enfoques tradicionales que dependen principalmente de índices predefinidos para determinar la importancia de los filtros. Los resultados obtenidos de este estudio son significativos. Con una tasa de compresión del 60%, lograron mantener una precisión del Top-1 de 0.8429 en el modelo VGG-16 bajo CIFAR-10, con una reducción de exactitud mínima de solo 0.0322.

Por otro lado, Ganjdanesh et al. [33] exponen una perspectiva diferente, argumentando que los métodos de XAI basados en gradiente, como DeepLIFT, LRP y *Maximation Activation*, no son confiables para la poda de modelos debido a que las características con gradientes más grandes no necesariamente tienen más influencia en el resultado del modelo. Es por esto que, para dirigir el proceso de poda de la red, utilizan interpretaciones del modelo en forma de mapas de relevancia (*saliency maps*), obtenidos a partir del método *Amortized Explanation Models* (AEMs). El desarrollo de su método incluyó la creación de un modelo selector que predice máscaras de saliencia en tiempo real, véase Figura 2.2, aplicando funciones similares a la Función de Base Radial (RBF). Esta aproximación facilita la obtención de explicaciones en formatos más compactos y reduce los recursos computacionales necesarios para la poda, ya que las máscaras de saliencia se generan en una sola pasada hacia adelante mediante un modelo selector. Además, al parametrizar cada máscara únicamente con tres valores se evita la necesidad de calcular distribuciones independientes para cada píxel, lo que disminuye drásticamente el número de operaciones requeridas respecto a métodos de interpretación tradicionales.

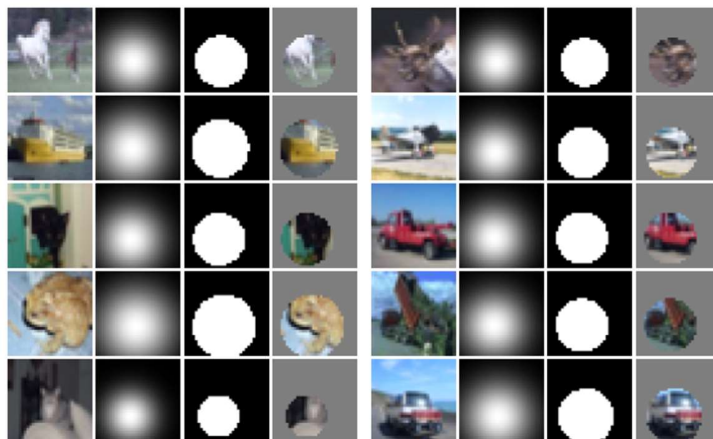


Figura 2.2. Mapas de saliencia generados con AEM [33].

Los resultados mostraron que este enfoque no solo logra reducir significativamente la complejidad y el tamaño de las CNN sin afectar negativamente su rendimiento, sino que también lo mejora. En experimentos extensivos realizados en conjuntos de datos estándar como CIFAR-10 e ImageNet, la metodología propuesta demostró su eficacia al lograr un rendimiento superior en comparación con los métodos base, además, muestra ventajas frente a los desempeños de otros métodos, incluyendo a QI [31], [32], como se observa en la Tabla 2.1.

Tabla 2.1. Resultados de los experimentos en ImageNet con ResNet-34/50/101 y MobileNet-V2 [33].

Model	Method	Baseline Top-1 Acc	Baseline Top-5 Acc	Δ -Acc Top-1	Δ -Acc Top-5	Pruned FLOPs
ResNet-34	FPGM	73.92%	91.62%	-1.29%	-0.54%	41.1%
	Taylor	73.31%	-	-0.48%	-	24.2%
	ISP	73.31%	91.42%	-0.45%	-0.40%	44.0%
ResNet-50	DCP	76.01%	92.93%	-1.06%	-0.61%	55.6%
	CCP	76.15%	92.87%	-0.94%	-0.45%	54.1%
	FPGM	76.15%	92.87%	-1.32%	-0.55%	53.5%
	ABCP	76.01%	92.96%	-2.15%	-1.27%	54.3%
	QI	74.90%	92.10%	-1.31%	-0.27%	50.0%
	PFP	76.13%	92.86%	-0.92%	-0.45%	44.0%
	EEMC	76.15%	92.87%	-0.49%	-0.35%	56.0%
	ISP	76.13%	92.86%	-0.16%	-0.12%	56.6%
ResNet-101	FPGM	77.37%	93.56%	-0.05%	0.00%	41.1%
	Taylor	77.37%	-	-0.02%	-	39.8%
	QI	76.40%	92.80%	-2.31%	-0.86%	50.0%
	PFP	77.37%	93.56%	-0.94%	-0.44%	45.1%
	ISP	77.37%	93.56%	+0.40%	+0.22%	56.8%
MobileNet-V2	Uniform	71.80%	91.00%	-2.00%	1.40%	30.0%
	AMC	71.80%	-	-1.00%	-	30.0%
	CC	71.88%	-	-0.97%	-	28.3%
	MetaPruning	72.00%	-	-0.80%	-	30.7%
	ISP	71.88%	90.29%	-0.15%	-0.08%	29.0%

A pesar de las limitaciones señaladas por Ganjdanesh et al. [33] sobre la poda de modelos con métodos basados en gradiente, el método de compresión de modelos utilizando LRP se ha

demostrado eficaz en diversas aplicaciones, destacando la eficiencia en la reducción de la complejidad sin comprometer el rendimiento esencial de los modelos de redes neuronales profundas. Luan et al. [34] aplicaron LRP para la poda de redes en agentes de aprendizaje por refuerzo profundo (DRL), mostrando que los agentes robustos entrenados con RADIAL-RL mantenían un mejor rendimiento post-poda comparado con los agentes estándar, especialmente en sistemas embebidos donde los recursos son limitados. Por otro lado, Ranjan y Savakis [35] utilizaron LRP para la cuantificación de transformadores visuales, asignando precisiones de bits mixtas basadas en la importancia de las capas determinada por LRP, logrando mejoras significativas en la precisión de clasificación de imágenes. Finalmente, Becking et al. [36] desarrollaron un enfoque de cuantificación guiado por explicabilidad, ECQx, que utiliza LRP para retener los pesos más importantes en la cuantización, mostrando resultados prometedores en la compresión de modelos manteniendo o incluso mejorando el rendimiento. Estos estudios subrayan la versatilidad y efectividad de LRP como herramienta para mejorar la eficiencia y comprensión de modelos de aprendizaje profundo en distintos contextos y arquitecturas.

2.3 Métodos de compresión en etapa de pre-entrenamiento

Mientras que las secciones anteriores se centraron en enfoques para la compresión de redes mediante técnicas aplicadas después del entrenamiento del modelo, es importante señalar que existen estrategias de compresión que modifican la estructura o arquitectura de un modelo, orillando a que estos cambios se realicen en etapas anteriores al entrenamiento de una red. En esta sección se busca explorar algunas de las técnicas más novedosas que caen en esta clasificación.

Menghani destaca que la eficiencia de un modelo puede ser optimizada para mejorar la inferencia [37]. El trabajo revisa técnicas de aprendizaje que alteran cómo se entrenan los modelos, tales como la destilación de conocimientos (*Knowledge Distillation*). Esta técnica permite que un modelo más pequeño absorba el conocimiento de un modelo más grande, beneficiando así la capacidad inferencial del modelo compacto.

Además, se aborda el área que explora arquitecturas eficientes, enfocándose en diseños basados en bloques fundamentales previamente establecidos. Ejemplos de esto incluyen las capas convolucionales, que han reemplazado a las capas completamente conectadas y los módulos de

atención que reemplazan a las redes neuronales recurrentes, buscando la eficiencia en el diseño desde el inicio.

Finalmente, Menghani señala la importancia de la automatización, como la Optimización de Hiperparámetros (*Hyper-Parameter Optimization*, HPO) y la Búsqueda de Arquitectura Neuronal (*Neural Architecture Search*, NAS). HPO usa algoritmos para encontrar de manera autónoma los hiperparámetros que mejoran el rendimiento de un modelo. NAS se enfoca en descubrir automáticamente arquitecturas que superen en rendimiento a aquellas diseñadas de forma manual, optimizando así tanto la calidad del modelo como su eficiencia en términos de tamaño y velocidad.

CAPÍTULO III. ENTORNO DE PRUEBAS

En este capítulo se muestran los detalles del entorno de pruebas que se utilizó para el desarrollo de esta tesis. El objetivo de este capítulo es mostrar las condiciones bajo las cuáles se realizaron los experimentos expuestos en esta tesis para asegurar la replicabilidad de los mismos, así como también lograr una comparación justa con trabajos anteriores y futuros.

3.1 Descripción de arquitecturas seleccionadas

Una arquitectura de red neuronal se entiende como la organización y disposición de sus componentes, lo que en conjunto define un espacio de funciones posibles que la red puede implementar. Esta arquitectura, por sí misma, no corresponde a una función concreta, sino que establece el marco dentro del cual pueden configurarse distintos modelos. Un modelo se obtiene al asignar valores específicos a los parámetros de la arquitectura. Los parámetros se modifican con un algoritmo de entrenamiento que utiliza un conjunto de datos D para ajustar el modelo y generar salidas de acuerdo con el *ground truth* del conjunto de datos. Formalmente, un modelo puede expresarse como $f(x; \mathbf{W})$, donde $\mathbf{W}$ representa los parámetros ajustados que determinan su comportamiento.

Parte importante del entorno de pruebas son las arquitecturas de los modelos profundos que se podrán. Se eligieron 4 diferentes arquitecturas con el propósito de comparar los resultados de poda en sus diferentes estructuras. Para elegir las arquitecturas se consideraron los siguientes aspectos:

- Arquitecturas usadas para visión por computadora que utilicen diferentes bloques/técnicas para resolver la tarea.
- Tener un buen rango de número de parámetros entre arquitecturas, es decir, considerar arquitecturas con gran y poca cantidad de parámetros.
- Que el desempeño de los modelos sea sobresaliente en la tarea para la que fueron diseñados.

Estas características permitirán evaluar en diferentes escenarios el impacto que generan las técnicas de compresión en términos de rendimiento y razón de compresión. Por otra parte, también

se buscó que las arquitecturas sean populares en la literatura con el objetivo de realizar comparaciones fácilmente y como indicativo de que dichas arquitecturas pueden ser reutilizadas y adaptadas a diferentes tareas con mayor facilidad, además, se seleccionaron aquellas que contaran con implementaciones en librerías populares y buena documentación.

Así pues, a continuación, se detallan las arquitecturas y se explica su estructura.

3.1.1 VGG19 con Batch Normalization

El modelo VGG-19, basado en la arquitectura de CNN que se propuso en el año 2014 [38] fue creado por el grupo Visual Geometry Group de la Universidad de Oxford para competir en el desafío de *ImageNet Large-Scale Visual Recognition Challenge* (ILSVRC) de ese año [39], cuyo objetivo era evaluar el desempeño de diferentes modelos en tareas de clasificación y detección de objetos a gran escala. La base de datos utilizada para este desafío contenía 1000 clases diferentes de objetos, lo cual representaba un reto considerable. Su principal aporte fue demostrar que incrementar la profundidad de la arquitectura, mediante la adición de más capas convolucionales respecto a diseños previos, podía mejorar significativamente el rendimiento en clasificación de imágenes.

Para lograr un buen desempeño en la tarea, VGG19 utilizó kernels con campos receptivos pequeños de 3x3 a lo largo de toda la red, a diferencia de arquitecturas anteriores que empleaban kernels de 5x5 o 7x7. La justificación de este enfoque es que dos capas convolucionales consecutivas con kernels de 3x3 tienen un campo receptivo equivalente a una capa de 5x5, mientras que tres capas con kernels de 3x3 son equivalentes a una de 7x7. En la Figura 3.1 se ilustra cómo dos capas de 3x3 logran el mismo campo receptivo que una sola capa de 5x5, mostrando cómo se combinan los campos receptivos pequeños para abarcar una mayor área. Esto permite introducir más funciones de activación no lineales entre las capas, lo que aumenta la capacidad del modelo para aprender y representar funciones complejas. Además, al utilizar kernels más pequeños, se logra reducir el número de parámetros necesarios, lo cual hace que la red sea más eficiente sin perder capacidad de representación.

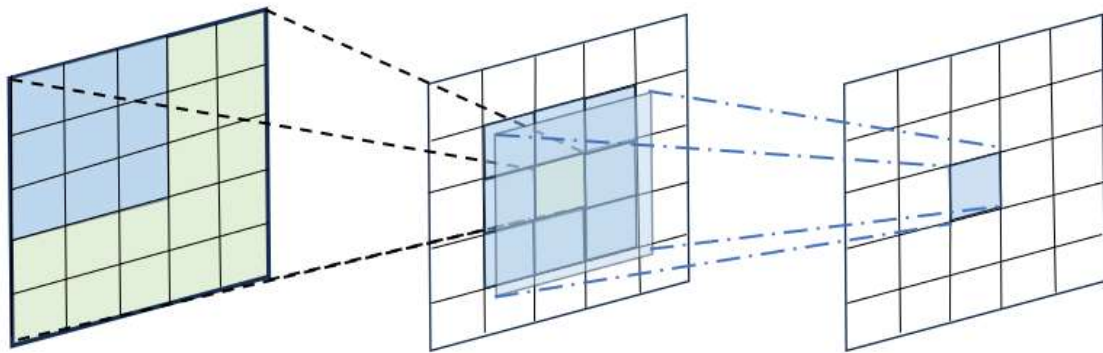


Figura 3.1. Comparación de campos receptivos de capas convolucionales con kernels de 3x3 (azul) y 5x5 (verde).

VGG-19 es una CNN con 19 capas de parámetros entrenables, de las cuales 16 son capas convolucionales y 3 son capas totalmente conectadas. La red está compuesta por bloques convolucionales, donde cada bloque tiene entre 2 y 4 capas convolucionales con kernels de 3x3. Todas las capas convolucionales usan un *stride* de 1, *padding* para mantener las mismas dimensiones, y la función de activación ReLU. Después de cada bloque convolucional, se incluye una capa de *max pooling* con ventana de 2x2 y un *stride* de 2, lo que reduce las dimensiones espaciales a la mitad.

Después de los 5 bloques convolucionales y sus respectivas capas de *max pooling*, se agregan 3 capas totalmente conectadas. Las dos primeras capas completamente conectadas tienen 4096 neuronas cada una, mientras que la última tiene tantas neuronas como clases objetivo, siendo 1000 para el caso de ImageNet. En trabajos posteriores [40], se creó una variación de la arquitectura de VGG19 en la que se añadieron capas de Batch Normalization (VGG19-BN) entre cada capa convolucional y la función de activación, permitiendo utilizar tasas de aprendizaje más altas y mejorando la capacidad de generalización del modelo. En la Figura 3.2 se muestra un diagrama con la secuencia de capas que conforman la arquitectura de VGG19-BN y las dimensiones de los tensores intermedios.

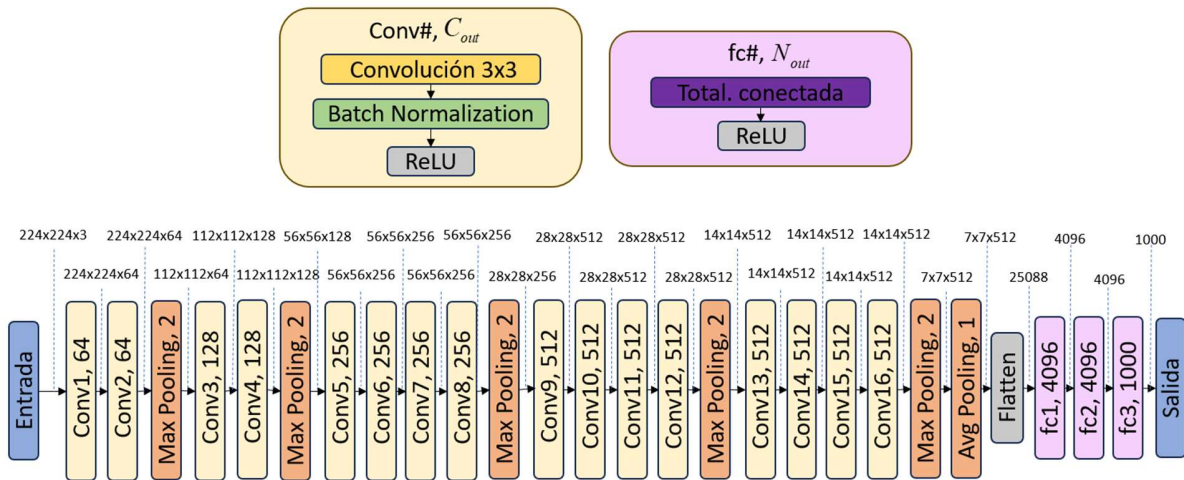


Figura 3.2. Arquitectura de VGG19-BN.

3.1.2 ResNet-18

Las arquitecturas de redes neuronales convolucionales ResNet-18 y sus variantes fueron creadas en el año 2015. ResNet-18 se destacó como una versión más eficiente computacionalmente en comparación con variantes más profundas como ResNet-152, lo cual la hizo más adecuada para aplicaciones con recursos limitados. Al igual que las redes VGG, participaron en el concurso ILSVRC, pero en la edición de 2015. Estas redes fueron desarrolladas en el laboratorio de Microsoft Research y la versión de ResNet con 152 capas ganó el primer lugar, logrando un error top-5 del 3.57% en ImageNet, lo cual representó un avance significativo en el campo de la clasificación de imágenes [41].

El objetivo principal de las redes ResNet fue abordar los problemas relacionados con el entrenamiento de redes profundas, como la disminución del gradiente y la degradación del rendimiento cuando se aumentaba la profundidad de la red. Los desarrolladores observaron que, al agregar más capas a una arquitectura profunda, el rendimiento del modelo se degradaba en lugar de mejorar, lo que indicaba que el proceso de optimización se volvía más complicado con redes cada vez más profundas.

Bajo la premisa de que, si se agregan capas que mapean funciones identidad a un modelo menos profundo, el rendimiento debería ser similar al original (Figura 3.3), se propuso una solución mediante conexiones residuales. En la práctica, los optimizadores no lograban encontrar una solución óptima que hiciera que las capas mapearan una función de tipo identidad, lo cual resultaba

en una degradación del rendimiento en modelos profundos. Para solucionar este problema, las conexiones residuales permiten que la entrada de una capa se salte varias capas y se sume a la salida de otra capa, facilitando el entrenamiento de redes profundas.

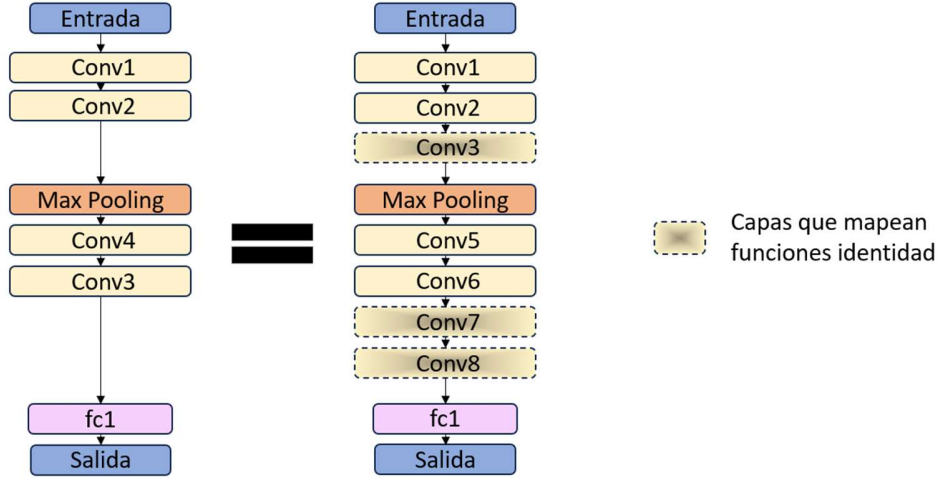


Figura 3.3. Agregación de capas que mapean funciones identidad.

Si se tiene una función objetivo $H(x)$ que se desea aproximar mediante varias capas no lineales apiladas, y siendo x la entrada a la primera capa. Se puede asumir que estas capas también pueden aproximar una función residual $F(x)$, definida como

$$F(x) = H(x) - x \quad (3.1)$$

De esta manera, en lugar de entrenar las capas para aproximar directamente $H(x)$, se entrena para aproximar $F(x)$ y se obtiene $H(x)$ sumando la entrada original:

$$H(x) = F(x) + x \quad (3.2)$$

El beneficio de aproximar $F(x)$ en lugar de $H(x)$ radica en que, si es necesario que $H(x)$ sea igual o similar a la función identidad ($H(x) = x$), es más sencillo entrenar la red para que la función residual sea cercana a cero, en lugar de entrenar para que $H(x)$ sea exactamente la función identidad.

Bajo este enfoque, se propuso utilizar conexiones residuales o de atajo entre capas, que permiten mapear la entrada x a la salida de otras capas, además de facilitar la propagación de los gradientes directamente hacia las capas iniciales sin que se degraden. Los bloques fundamentales utilizados en ResNet se denominan bloques residuales y están compuestos por dos capas convolucionales con kernels de 3×3 , cada una seguida por una capa de *Batch Normalization* y una función de activación ReLU. En los bloques donde se reduce la dimensionalidad espacial mediante *strides*, se utiliza una convolución de 1×1 para ajustar las dimensiones antes de realizar la suma. En la siguiente figura se muestra la estructura de este bloque en sus versiones con y sin *stride*.

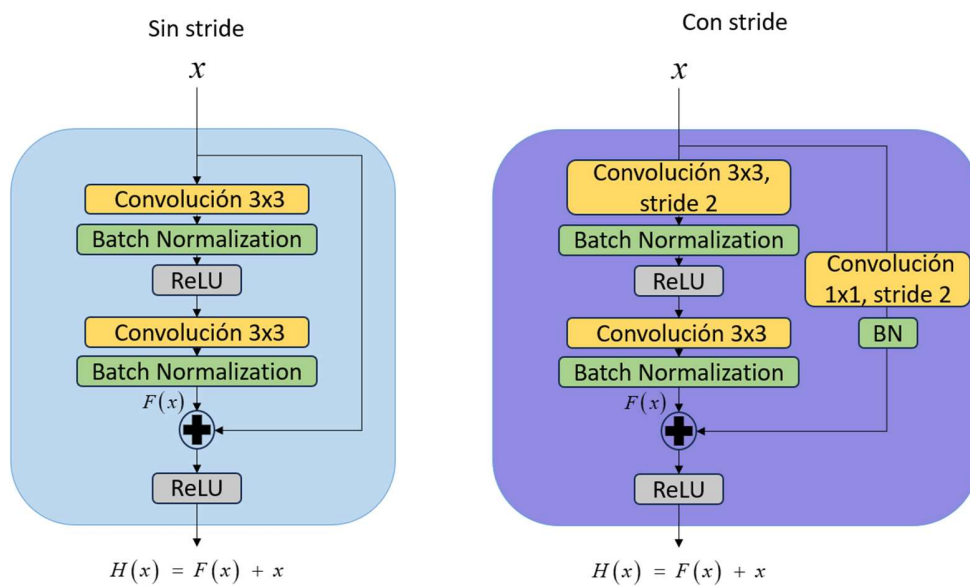


Figura 3.4. Estructura de bloques residuales sin y con stride.

El nombre de ResNet-18 proviene de las 18 capas entrenables que utiliza, distribuidas en bloques residuales. La arquitectura mostrada en la Figura 3.5 comienza con una capa convolucional de 7×7 seguida de una capa de *max pooling*. Luego, se incluyen cuatro bloques residuales, cada uno con dos bloques que contienen dos convoluciones de 3×3 con *Batch Normalization* y activaciones ReLU, sumando la entrada original al final mediante la conexión residual. En los bloques donde se reducen las dimensiones espaciales mediante *strides*, las conexiones residuales también ajustan las dimensiones utilizando una convolución 1×1 . Finalmente, la red termina con un *average pooling* global y una capa totalmente conectada para la clasificación.

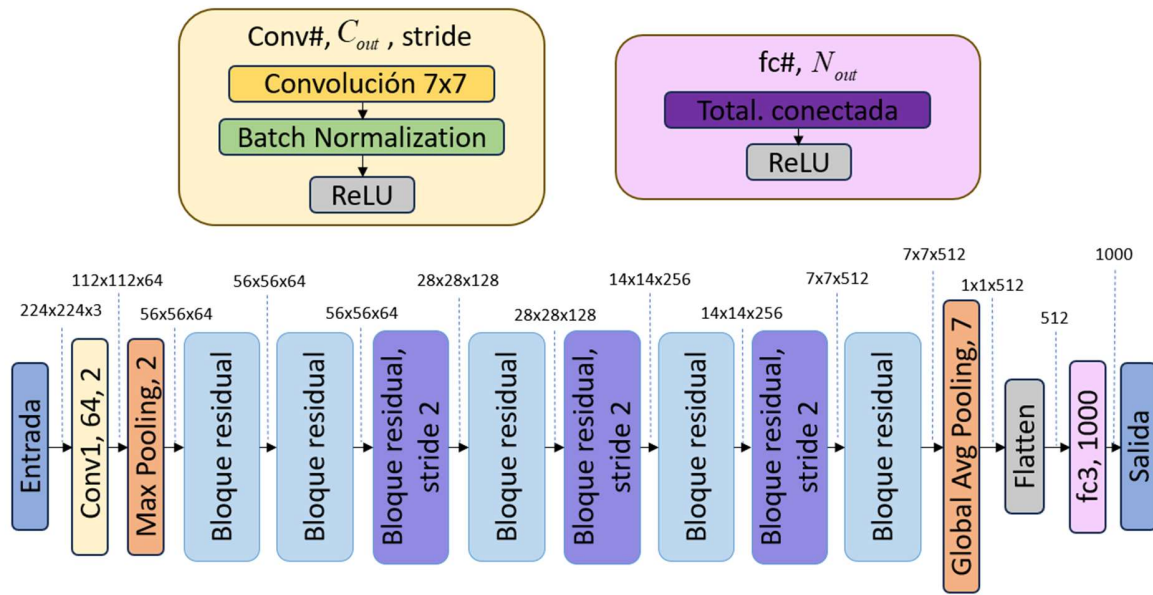


Figura 3.5. Arquitectura de ResNet-18.

3.1.3 ResNet-50

ResNet-50 es un CNN que fue desarrollada al mismo tiempo que ResNet-18, compartiendo el objetivo de facilitar el entrenamiento de redes profundas mediante el uso de conexiones residuales para evitar la degradación del rendimiento al aumentar la profundidad de la red. Funciona bajo el mismo enfoque de funciones de identidad y fue diseñada para validar que las conexiones residuales permiten crear arquitecturas más profundas [41].

Con el objetivo de reducir el tiempo de entrenamiento, se optó por modificar los bloques residuales utilizando un diseño de cuello de botella (*Bottleneck*), el cual reduce la cantidad de parámetros y, por lo tanto, disminuye el costo computacional durante el entrenamiento. Cada bloque de cuello de botella consta de tres capas, una convolución 1x1 que reduce la dimensionalidad para disminuir el número de parámetros, una convolución 3x3 que realiza la extracción de características y otra convolución 1x1 que restaura o expande la dimensionalidad para recuperar la capacidad de representación del modelo. Las conexiones de atajo suman la entrada original a la salida del bloque y cuando hay cambios en las dimensiones espaciales o en los canales, se utiliza una convolución 1x1 en el atajo para ajustar las dimensiones. En la siguiente imagen se detalla esta modificación de los bloques.

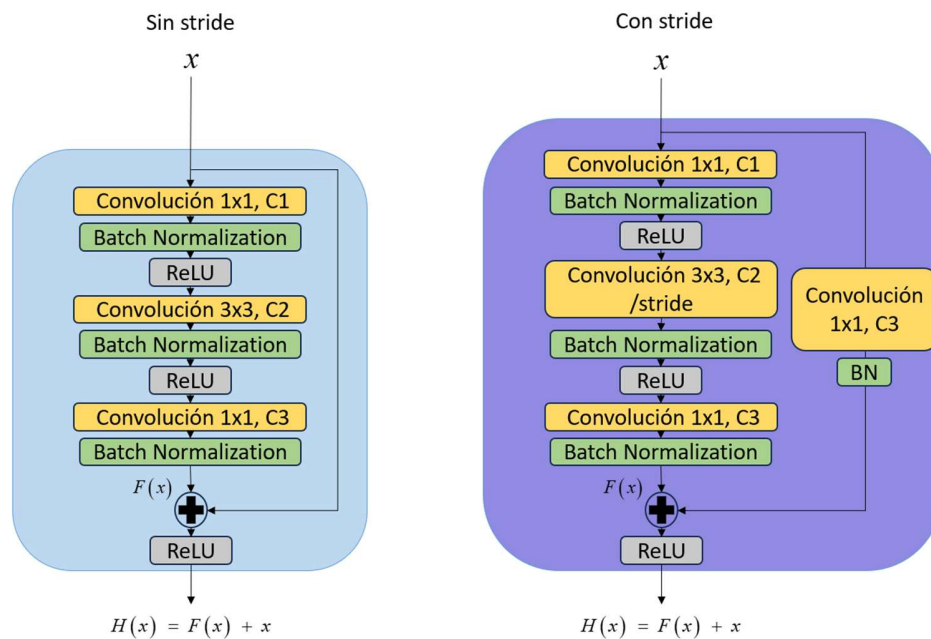


Figura 3.6. Bloques residuales con cuello de botella.

Ahora bien, la arquitectura de ResNet-50 utiliza los bloques residuales tipo cuello de botella en lugar de los bloques residuales convencionales, haciendo que la red cuente con 50 capas entrenables, lo cual permite mejorar la extracción de características y la precisión del modelo. Su arquitectura comienza con una convolución 7x7 seguida de *max pooling*. Luego, incluye cuatro etapas de bloques residuales con 3, 4, 6 y 3 bloques, respectivamente, sumando un total de 16 bloques residuales. Los canales aumentan progresivamente de 64 a 2048 a través de las etapas y el tamaño espacial se reduce mediante pasos en las convoluciones. La red finaliza con un *average pooling* global y una capa totalmente conectada para la clasificación, como se muestra en la Figura 3.7.

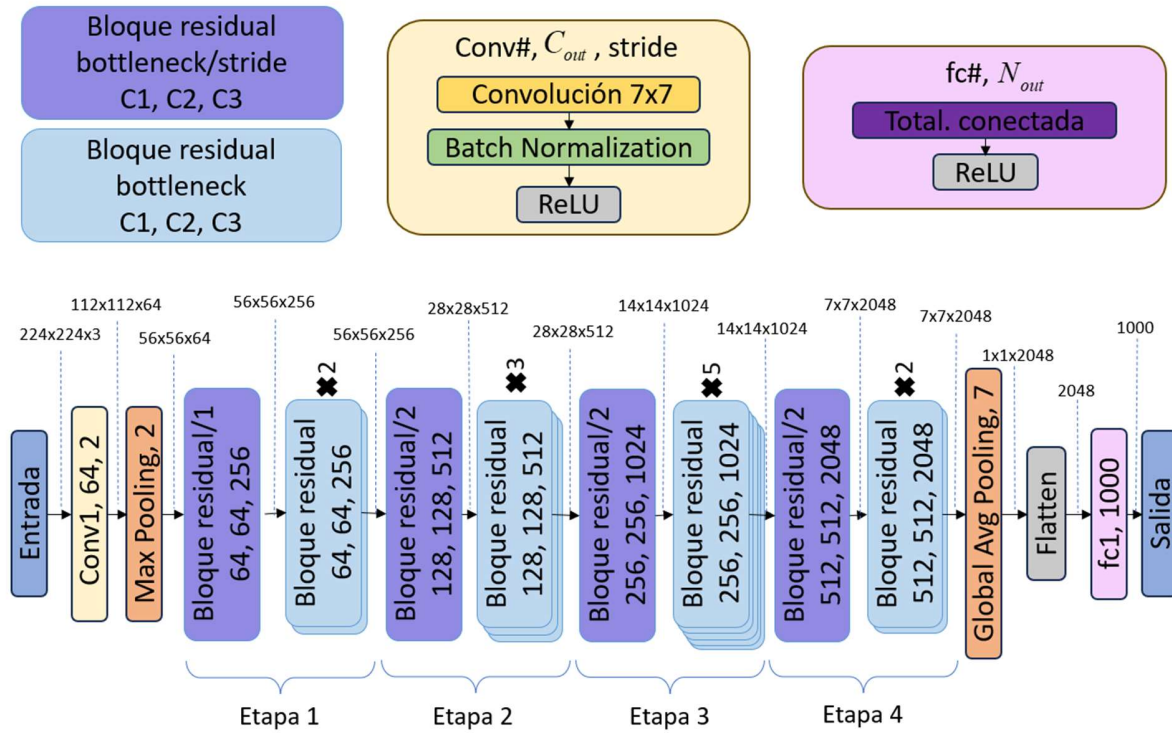


Figura 3.7. Arquitectura de ResNet-50.

3.1.4 DenseNet-121

DenseNet-121 [42] fue introducida en el año 2016, y su nombre, DenseNet, proviene del esquema de realizar muchas conexiones entre capas, lo que resulta en una gran densidad de conexiones. El 121 en el nombre se refiere al número total de capas de la red. DenseNet continúa los descubrimientos hechos para ResNet [41], como el uso de conexiones para facilitar el entrenamiento de redes profundas, al proponer un método que fortalece la propagación de las características a lo largo de la red.

El objetivo principal de DenseNet es solucionar el problema del desvanecimiento de gradientes y reducir el número de parámetros. Para lograr esto, se fortalece la propagación de características mediante una gran cantidad de conexiones, donde cada capa se conecta directamente con todas las capas anteriores. A diferencia de ResNet, DenseNet no utiliza conexiones residuales aditivas, es decir, la entrada de una capa no se suma con la salida de las siguientes capas. En su lugar, las entradas se concatenan con las salidas, lo que permite reusar las características extraídas por capas anteriores y mitigar el desvanecimiento de gradientes. La gran cantidad de conexiones facilita el

entrenamiento, ya que cada capa tiene acceso directo tanto a los gradientes de la función de pérdida como a la entrada original.

En una red ResNet, si la capa ℓ aproxima una función $H_\ell(\cdot)$, la salida de esta capa se suma con la entrada original (ecuación 3.1) para obtener la entrada de la siguiente capa $x_{\ell+1}$.

$$x_{\ell+1} = H_{\ell+1}(x_\ell) + x_\ell \quad (3.3)$$

Sin embargo, en DenseNet, en lugar de sumar las entradas con las salidas, todas las entradas de capas anteriores se concatenan con la entrada de la capa actual. Como resultado, la entrada de la capa $\ell+1$ sería la concatenación de los mapas de características de las capas anteriores.

$$x_{\ell+1} = H_{\ell+1}([x_0, x_1, x_2, \dots, x_\ell]) \quad (3.4)$$

Esta concatenación solo es posible cuando los mapas de características tienen la misma dimensión. Por lo tanto, para aprovechar los beneficios de submuestrear el tamaño de los mapas de características que proporcionan las CNN y utilizar el enfoque de conexiones densas, DenseNet utiliza los llamados bloques densos. Cada bloque denso consta de varias capas, donde se aplica un diseño de cuello de botella para reducir el número de mapas de características y mejorar la eficiencia computacional. Este diseño de cuello de botella consiste en utilizar una capa de convolución 1x1 para reducir la dimensionalidad, seguida de una convolución 3x3. Antes de cada capa convolucional se cuenta con capas de Batch Normalization y función de activación ReLU. En cada capa del bloque denso, se producen k canales, y para producir la entrada de la capa ℓ se concatenan los mapas de características de todas las capas anteriores. De esta forma, la ℓ -ésima capa dentro de un bloque denso tendrá $k_0 + k(\ell - 1)$ mapas de características de entrada, donde k es conocido como la tasa de crecimiento (*growth rate*). En la Figura 3.8 se ilustra un bloque denso.

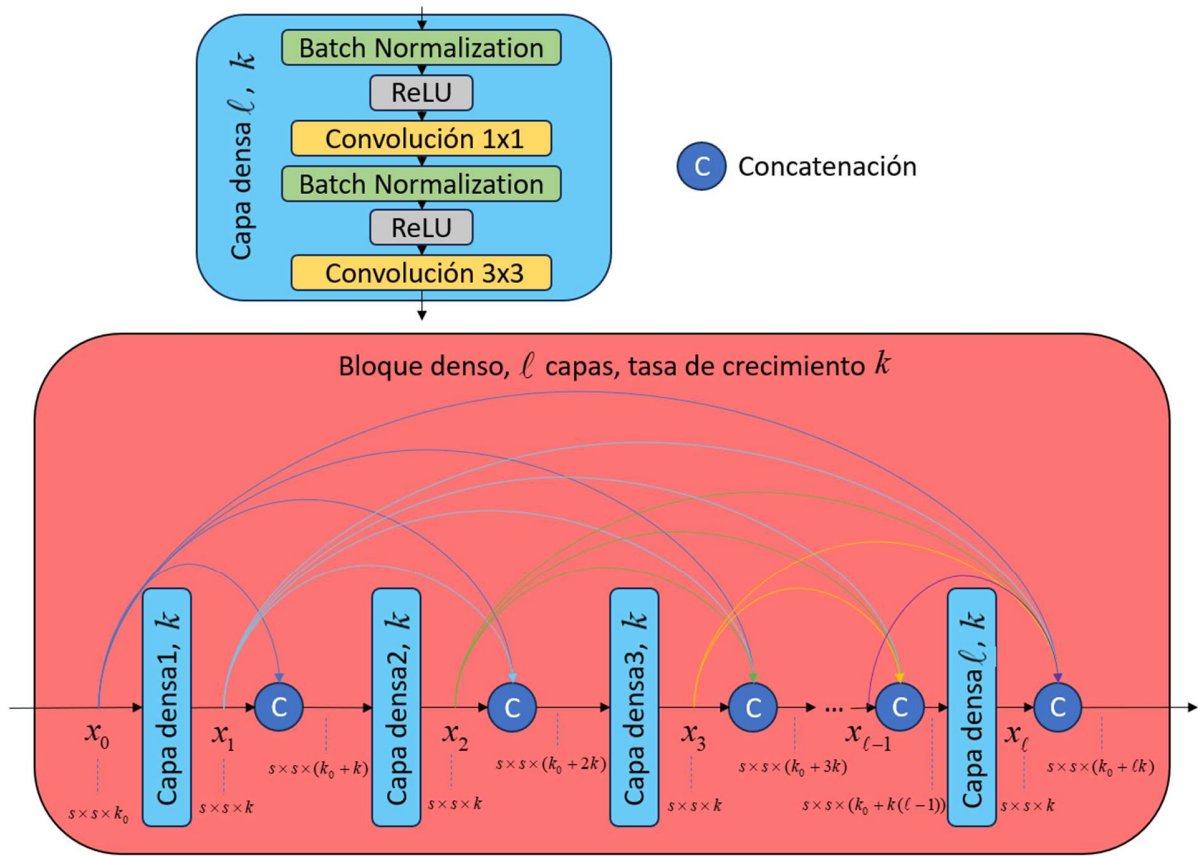


Figura 3.8. Estructura de bloque denso.

Las capas que se encuentran entre los bloques densos se encargan de reducir la dimensionalidad de los mapas de características mediante capas de *average pooling*. Estas capas de transición permiten mantener el tamaño de la red manejable y mejorar la eficiencia computacional de la arquitectura. Estas capas de transición están compuestas como se muestra a continuación.

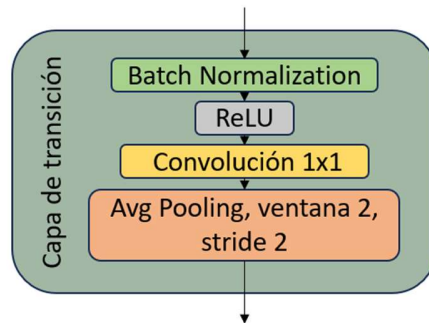


Figura 3.9. Estructura de capa de transición.

La arquitectura de DenseNet-121 comienza con una capa de convolución de 7×7 con 64 canales, seguida de una capa de *Batch Normalization* y una función de activación ReLU. A continuación, se aplica una capa de *max pooling* con un *stride* de 2. Luego, se implementan cuatro bloques densos separados por capas de transición. Estos bloques densos contienen 6, 12, 24 y 16 capas, respectivamente. Después del último bloque denso, la red aplica una capa de *average pooling* global y concluye con una capa totalmente conectada para realizar la clasificación final, como se muestra en la Figura 3.10.

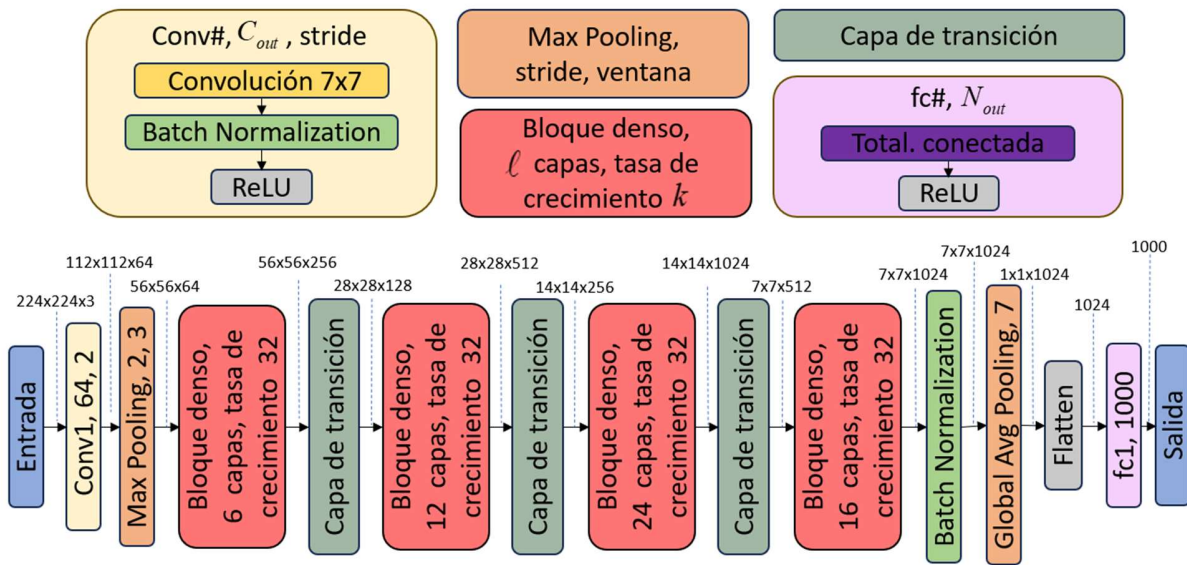


Figura 3.10. Arquitectura de DenseNet-121.

3.2 Conjuntos de datos utilizados

Se plantea utilizar tres conjuntos de datos populares en la literatura con el propósito de validar el comportamiento de los métodos de poda bajo diferentes contextos que permitan evaluar si las variaciones en los conjuntos de datos impactan en los métodos de poda, observar si los métodos de poda logran conservar la capacidad de los modelos de adaptarse a bases de datos con diferentes complejidades y asegurarse que los resultados obtenidos con un modelo específico no se deban a las características de un conjunto de datos específico.

3.2.1 ImageNet-1K

ImageNet consiste en 14,197,122 de imágenes de resolución variable que se categorizan en 21,841 clases correspondientes a sustantivos provenientes de la jerarquía WordNet, una base de

datos léxicos del idioma inglés desarrollada en la Universidad de Princeton donde las palabras se agrupan según sus significados. Cada grupo se denomina *synset* (conjunto sinónimo), es decir, cada grupo contiene palabras que son sinónimos entre sí. Así pues, cada clase en ImageNet corresponde a un *synset* de WordNet relacionados con objetos que se pueden reconocer visualmente, como animales, vehículos, herramientas, plantas, entre otros [43].

El primer conjunto de datos seleccionado se denomina ImageNet-1K, es un subconjunto de la base de datos ImageNet que fue utilizado para la competencia anual de clasificación y detección de imágenes ILSVRC [39], a partir de la cual fue ganando popularidad. Este subconjunto consiste en 1000 clases distribuidas en 1,281,167 imágenes de entrenamiento, 50,000 de validación y 100,000 de prueba. El conjunto de entrenamiento presenta un ligero desbalance, ya que el número de imágenes por clase varía entre aproximadamente 700 y 1,300. En contraste, los conjuntos de validación y prueba están balanceados, con 50 y 100 imágenes por clase, respectivamente. En la Figura 3.11 se muestran ejemplos de imágenes pertenecientes a esta base de datos.



Figura 3.11. Ejemplos de imágenes pertenecientes a las clases *hamster*, *zebra*, *chain* y *corn* del conjunto de datos ImageNet-1K.

Las arquitecturas presentadas en la sección anterior fueron diseñadas para trabajar con el conjunto de datos ImageNet-1K. Por consiguiente, las arquitecturas no requieren de modificaciones para su funcionamiento con esta base de datos. Sin embargo, debido a la variedad en resoluciones de las imágenes, es necesario realizar un redimensionamiento y preprocesamiento en ellas para adaptarlas al entrenamiento e inferencia de una DNN.

El preprocesamiento de las imágenes sigue una serie de pasos para garantizar la consistencia en el tamaño y las características de las entradas. En primer lugar, las imágenes se redimensionan a un tamaño de 256 píxeles usando interpolación bilineal porque proporciona un equilibrio adecuado entre suavidad y preservación de detalles, evitando artefactos que podrían afectar el aprendizaje

del modelo, y mantiene un bajo costo computacional durante el preprocesamiento. Luego, se aplica un recorte central de tamaño 224 píxeles para asegurar un tamaño uniforme de entrada. Posteriormente, los valores de las imágenes se escalan al rango $[0.0, 1.0]$ y se normalizan utilizando las medias $[0.485, 0.456, 0.406]$ y desviaciones estándar $[0.229, 0.224, 0.225]$, correspondientes a los valores calculados para el conjunto de datos ImageNet-1K. En la siguiente figura se muestran las imágenes de la Figura 3.11 con preprocesamiento descrito.



Figura 3.12. Imágenes pertenecientes a ImageNet-1K de las clases *hamster*, *zebra*, *chain* y *corn* preprocesadas.

3.2.2 CIFAR-10

CIFAR-10 es un conjunto de datos de 60,000 imágenes a color de resolución de 32×32 píxeles, la base de datos se divide en 10 clases y a cada clase le corresponde 6,000 imágenes [44]. El conjunto de entrenamiento consiste de 50,000 imágenes y el conjunto de validación es de 10,000 imágenes. CIFAR-10 es a su vez un subconjunto del conjunto de datos *80 Million Tiny Images*.

Las 10 clases de este conjunto de datos consisten en *airplane*, *automobile*, *bird*, *cat*, *deer*, *dog*, *frog*, *horse*, *ship* y *truck*. En la Figura 3.13 se muestran 10 ejemplos de cada una de estas clases.

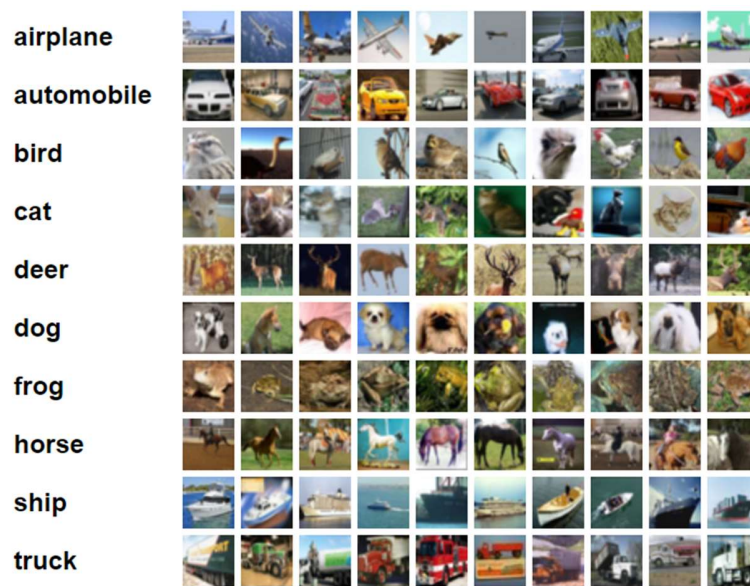


Figura 3.13. Ejemplos de imágenes de las 10 clases de CIFAR-10 [45].

Debido a que la resolución de las imágenes es considerablemente menor a las imágenes de ImageNet-1K, es necesario adaptar las arquitecturas de las DNN para lograr buenos desempeños. Cabe resaltar que, al clasificar solamente 10 clases, las capas totalmente conectadas finales de todas las DNN se configuran para que cuente con 10 neuronas de salida. A continuación, se enlistan los cambios específicos realizados en cada una de las arquitecturas.

- VGG19-BN:

Al tener como entrada de la red una imagen de menor resolución, la dimensionalidad de los mapas de características también se ve reducida, por lo que se modifica la capa de *Average Pooling* para que las dimensiones de su salida resulten en $1 \times 1 \times 512$, en lugar del tamaño original de $7 \times 7 \times 512$. La arquitectura y las dimensiones de los mapas de características resultantes se observan en la Figura 3.14.

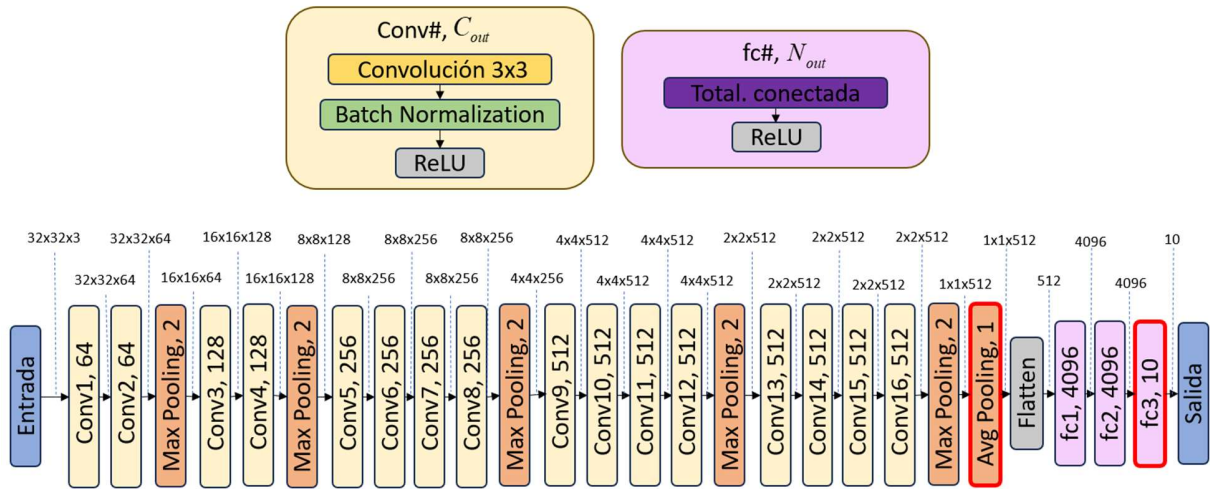


Figura 3.14. Arquitectura de VGG19-BN modificada para utilizarse con CIFAR-10.

- ResNet-18/ResNet-50/DenseNet-121

Para las arquitecturas de ResNet-18, ResNet-50 y DenseNet-121 se ajusta la primera capa convolucional donde cambia el tamaño del *kernel* de 7x7 a 3x3 para adaptarse mejor a imágenes de baja resolución y el *stride* se reduce a 1 para mantener más información de las imágenes pequeñas. Los cambios se ven reflejados la Figura 3.15, Figura 3.16 y Figura 3.17.

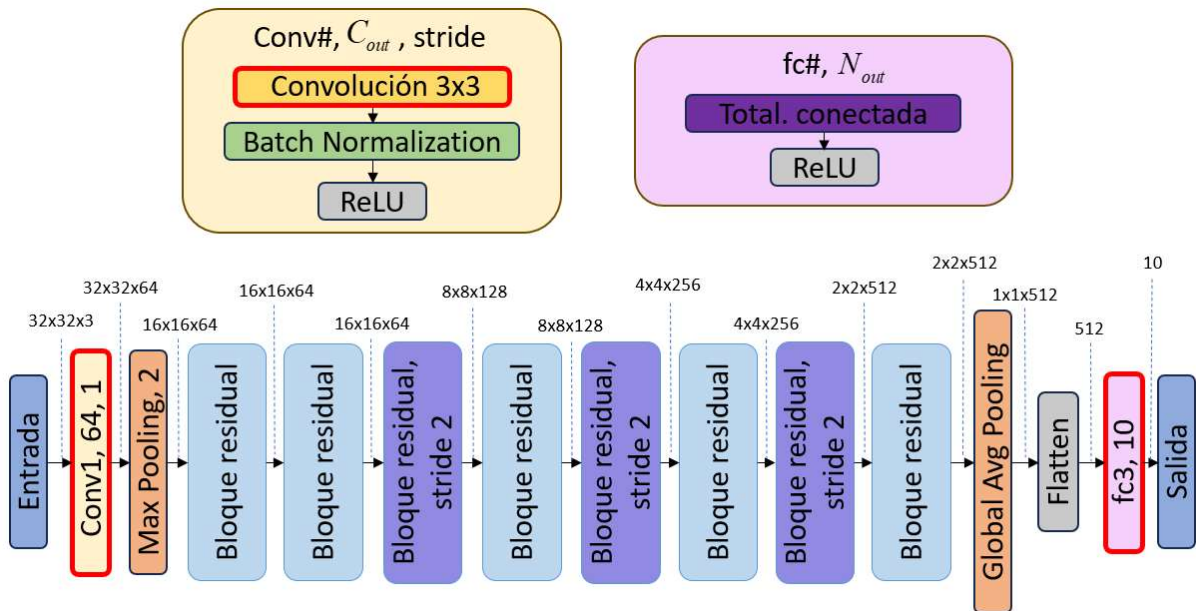


Figura 3.15. Arquitectura de ResNet-18 modificada para utilizarse con CIFAR-10.

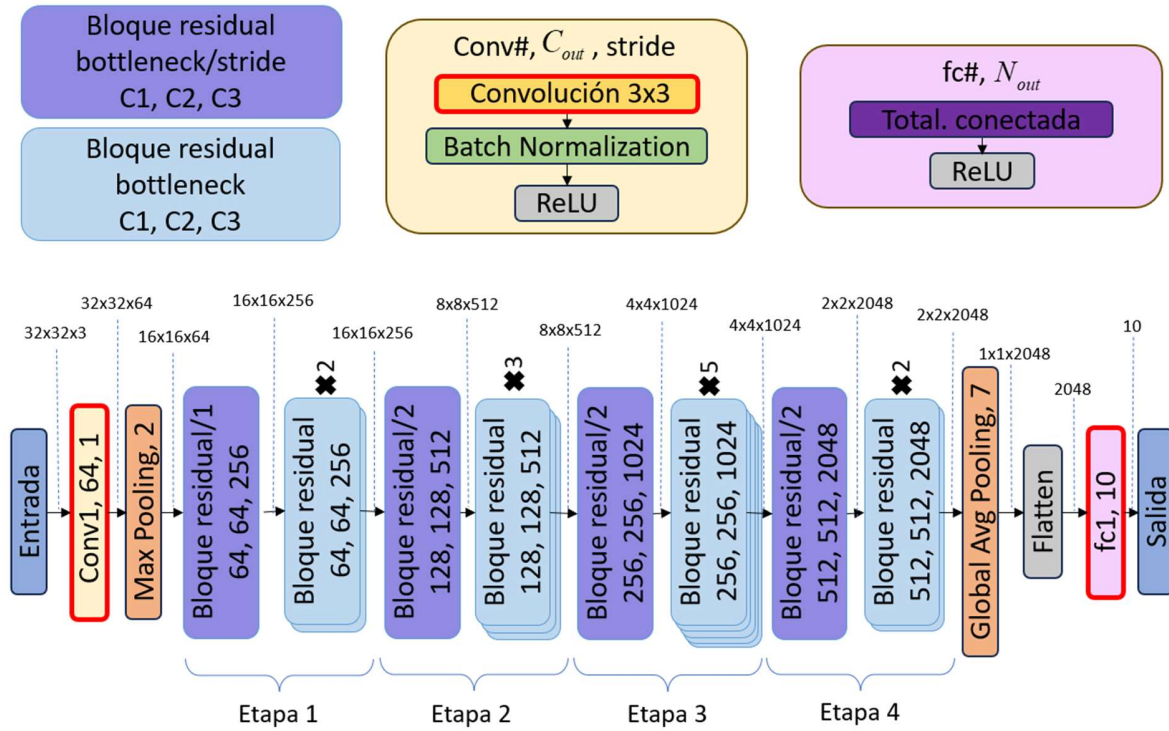


Figura 3.16. Arquitectura de ResNet-50 modificada para utilizarse con CIFAR-10.

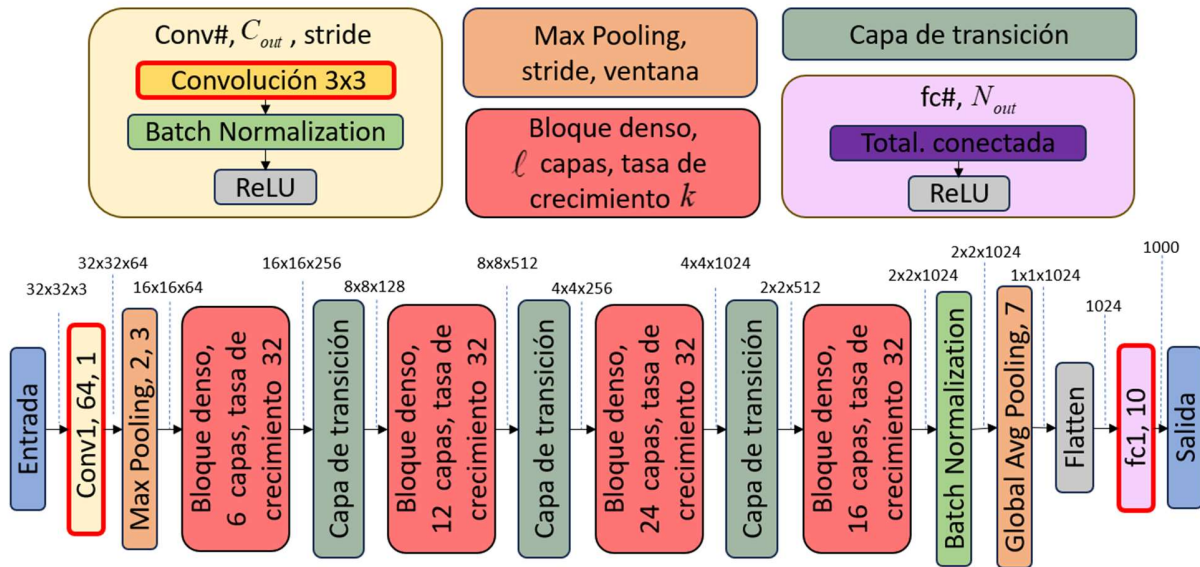


Figura 3.17. Arquitectura de DenseNet-121 modificada para utilizarse con CIFAR-10.

Las imágenes también pasan por un preprocesamiento que incluye la normalización utilizando las medias $[0.4914, 0.4822, 0.4465]$ y desviaciones estándar $[0.2471, 0.2435, 0.2616]$ para CIFAR-10.

3.2.3 CIFAR-100

Al igual que CIFAR-10, el conjunto de datos CIFAR-100 es un subconjunto de la base de datos *80 Million Tiny Images*. CIFAR-100 consiste de 100 clases organizadas en 20 superclases, cada una agrupando clases relacionadas. Cada imagen tiene dos tipos de etiquetas, una etiqueta detallada que especifica la clase particular de la imagen, y una etiqueta más general que identifica la superclase a la que pertenece, agrupando clases relacionadas. En la Tabla 3.1 se enlistan las clases que corresponde a cada superclase. Cada clase contiene 600 imágenes a color de 32x32 píxeles, de las cuales 500 se utilizan para entrenamiento y 100 para validación. En total, el conjunto de datos cuenta con 60,000 imágenes [44].

Debido a que las imágenes en CIFAR-100 tienen la misma resolución que las imágenes de CIFAR-10, no es necesario hacer ajustes adicionales a las arquitecturas modificadas mostradas en la sección 3.2.2. A excepción de cambiar a 100 el número de neuronas de salida de la capa totalmente conectada final para la obtener los resultados de clasificación de las 100 clases.

El preprocesamiento de las imágenes para realizar el entrenamiento e inferencia de la arquitectura es similar al utilizado para el conjunto de datos CIFAR-10, sin embargo, para CIFAR-100 se utilizan las medias [0.5071, 0.4867, 0.4408] y las desviaciones estándar [0.2675, 0.2565, 0.2761].

Tabla 3.1. Superclases y clases de CIFAR-100 [45].

Superclase	Clases
<i>aquatic mammals</i>	<i>beaver, dolphin, otter, seal, whale</i>
<i>fish</i>	<i>aquarium fish, flatfish, ray, shark, trout</i>
<i>flowers</i>	<i>orchids, poppies, roses, sunflowers, tulips</i>
<i>food containers</i>	<i>bottles, bowls, cans, cups, plates</i>
<i>fruit and vegetables</i>	<i>apples, mushrooms, oranges, pears, sweet peppers</i>
<i>household electrical devices</i>	<i>clock, computer keyboard, lamp, telephone, television</i>
<i>household furniture</i>	<i>bed, chair, couch, table, wardrobe</i>
<i>insects</i>	<i>bee, beetle, butterfly, caterpillar, cockroach</i>
<i>large carnivores</i>	<i>bear, leopard, lion, tiger, wolf</i>
<i>large man-made outdoor things</i>	<i>bridge, castle, house, road, skyscraper</i>
<i>large natural outdoor scenes</i>	<i>cloud, forest, mountain, plain, sea</i>
<i>large omnivores and herbivores</i>	<i>camel, cattle, chimpanzee, elephant, kangaroo</i>
<i>medium-sized mammals</i>	<i>fox, porcupine, possum, raccoon, skunk</i>
<i>non-insect invertebrates</i>	<i>crab, lobster, snail, spider, worm</i>
<i>people</i>	<i>baby, boy, girl, man, woman</i>
<i>reptiles</i>	<i>crocodile, dinosaur, lizard, snake, turtle</i>
<i>small mammals</i>	<i>hamster, mouse, rabbit, shrew, squirrel</i>
<i>trees</i>	<i>maple, oak, palm, pine, willow</i>
<i>vehicles 1</i>	<i>bicycle, bus, motorcycle, pickup truck, train</i>
<i>vehicles 2</i>	<i>lawn-mower, rocket, streetcar, tank, tractor</i>

3.3 Listado de modelos e hiperparámetros

Por cada conjunto de datos seleccionado, se genera un modelo distinto por cada arquitectura elegida, adaptando los parámetros de la arquitectura a los datos específicos para optimizar el ajuste

del modelo. Esto implica que, para cada base de datos, se entrenarán 3 modelos diferentes de cada arquitectura.

En la literatura relacionada a métodos de poda [10], se describe un flujo de trabajo comúnmente utilizado por muchos algoritmos de poda, el cual consta de tres etapas: (1) preentrenamiento, en el cual se entrena un modelo inicial, (2) poda, en la cual se eliminan pesos o neuronas del modelo inicial siguiendo ciertas reglas y (3) *finetuning*, donde el modelo podado se re-entrena para recuperar el rendimiento perdido. Tomando en cuenta esto, se propone el principio de equidad, que consiste en que la ventaja de rendimiento de un algoritmo de poda debe atribuirse a la etapa de poda, no al *finetuning* ni al entrenamiento previo [10].

En consecuencia, se establece que, para lograr una comparación justa entre algoritmos de poda, es preferible contar con las mismas condiciones iniciales, esto es utilizar los mismos modelos pre-entrenados de cada arquitectura a los cuales se les aplicará la poda.

Entre los modelos utilizados para realizar experimentos de poda se encuentran los modelos pre-entrenados con ImageNet-1k proporcionados por la librería de PyTorch [46]. Mientras que los modelos entrenados con CIFAR-10 fueron obtenidos de un repositorio de GitHub [47], dando prioridad a repositorios que faciliten la reproducibilidad de los experimentos. Por otro lado, debido a la falta de repositorios con modelos pre-entrenados en CIFAR-100, fue necesario llevar a cabo el entrenamiento personalizado de estos modelos.

A continuación, se detallan los hiperparámetros empleados durante el proceso de entrenamiento para cada base de datos en la Tabla 3.2.

Se obtuvieron un total de 12 modelos, en la Tabla 3.3 se resumen los desempeños obtenidos, expresados en términos de *accuracy*, para estas DNNs al ser entrenadas y evaluadas con sus respectivas bases de datos. En el caso de los modelos entrenados con ImageNet-1K también se reporta el *accuracy* top-5, que representa la proporción de casos en los que la categoría correcta se encuentra entre las cinco predicciones más probables. El *accuracy* es una métrica suficiente para comparar desempeño global, porque cada clase tiene la misma cantidad de ejemplos en los conjuntos de datos CIFAR-10 y CIFAR-100, mientras que el desbalance en ImageNet-1K no es extremo.

Tabla 3.2. Hiperparámetros para el entrenamiento de las arquitecturas.

Hiperparámetro	CIFAR-10	CIFAR-100	ImageNet-1K
Tamaño de lote	256	256	32
Épocas	100	100	90
Optimizador	SDG con Nesterov Momentum (0.9)	SDG con Nesterov Momentum (0.9)	SDG con Nesterov Momentum (0.9)
Tasa de aprendizaje inicial	0.01	0.01	0.1
Programa de tasa de aprendizaje	Ajuste cosenoidal con fase de calentamiento	Ajuste cosenoidal con fase de calentamiento	Se reduce a una décima parte cada 30 épocas
Weight decay	0.01	0.01	0.0001

Tabla 3.3. Accuracy de los modelos pre-entrenados.

	ResNet-18	ResNet-50	VGG19-BN	DenseNet-121
ImageNet-1K	69.75% / 89.07%	76.13% / 92.86%	74.21% / 91.84%	74.43% / 91.97%
CIFAR-10	92.59%	92.78%	93.28%	93.17%
CIFAR-100	71.58%	73.52%	71.28%	74.36%

Estos modelos servirán como modelos base sobre los cuales se realizará la compresión de modelos.

3.4 Métricas de evaluación

En cuanto a las métricas empleadas para evaluar los experimentos, se clasifican en dos categorías: métricas para evaluar la compresión del modelo y una métrica para medir el desempeño predictivo de las redes neuronales profundas.

3.4.1 Métricas de compresión

La evaluación de la compresión de los modelos se realiza mediante las siguientes métricas.

- Porcentaje de reducción de parámetros: Esta métrica mide la reducción relativa en el número de parámetros del modelo comprimido en comparación con el modelo original. Se define mediante la siguiente expresión:

$$\% \downarrow \text{parms.} = \left(1 - \frac{\text{No. de parms. de mod. comprimido}}{\text{No. de parms. de mod. original}} \right) * 100\% \quad (3.5)$$

- Porcentaje de reducción de FLOPs: Esta métrica evalúa la reducción en las operaciones de punto flotante (FLOPs) necesarias para ejecutar el modelo comprimido. Se calcula como:

$$\% \downarrow \text{FLOPs} = \left(1 - \frac{\text{No. de FLOPs de mod. comprimido}}{\text{No. de FLOPs de mod. original}} \right) * 100\% \quad (3.6)$$

- Aceleración: Esta métrica indica cuánto más rápido es el modelo comprimido en términos de operaciones de punto flotante en comparación con el modelo original. Se expresa mediante:

$$\text{Aceleración} = \frac{\text{No. de FLOPs de modelo original}}{\text{No. de FLOPs de modelo comprimido}} \quad (3.7)$$

3.4.2 Métrica de desempeño

Para evaluar el desempeño predictivo de las redes neuronales, se utiliza la métrica de *accuracy*, la cual mide la proporción de predicciones correctas sobre el total de predicciones realizadas. Su definición formal es:

$$\text{Accuracy} = \frac{1}{N} \sum_{i=1}^N I(\hat{y}_i = y_i) \quad (3.8)$$

Donde N es el número total de muestras, $\hat{y}_i$ es la clase predicha para la muestra, y_i la clase verdadera para la muestra y $I(\hat{y}_i = y_i)$ toma el valor de 1 si la clase predicha coincide con la verdadera y 0 en caso contrario.

3.5 Frameworks y librerías

En esta sección se describen los *frameworks* y librerías más relevantes empleados en el desarrollo de esta investigación. Estas herramientas fueron seleccionadas debido a su amplia adopción en la comunidad de ML, su documentación y su capacidad para facilitar la implementación de modelos y técnicas.

- PyTorch

PyTorch es un *framework* de aprendizaje profundo ampliamente utilizado por su flexibilidad y facilidad de uso [46]. En esta investigación, PyTorch se empleó para:

- Construir arquitecturas de las redes neuronales profundas.
 - Entrenar y evaluar modelos mediante el manejo de tensores y compatibilidad con GPU.
 - Implementar funciones de optimización, programadores de tasa de aprendizaje y cálculo de la función de pérdida.
- Torch-Pruning

Torch-Pruning es un *framework* especializado en la implementación de métodos de poda estructural para redes neuronales [48]. Esta herramienta utiliza un algoritmo llamado DepGraph utilizado para agrupar parámetros dependientes entre sí y posteriormente removerlos, a diferencia de otras implementaciones que establecen en cero los pesos en lugar de eliminarlos físicamente. Torch-Pruning fue utilizado para; aplicar diferentes métodos de poda de forma estructural, calcular la reducción en el número de parámetros y FLOPs tras la poda, y evaluar el rendimiento de los modelos comprimidos.

- Matplotlib

Matplotlib es una librería de visualización que permite generar gráficas y figuras. En este trabajo se utilizó para visualizar las curvas de entrenamiento y validación.

- argparse

La librería argparse facilita la configuración de hiperparámetros y parámetros de ejecución desde la línea de comandos. En este trabajo, argparse se utilizó para definir opciones de ejecución como el modo de operación, conjunto de datos y arquitectura y también fue utilizada para configurar hiperparámetros como la tasa de aprendizaje, el tamaño del lote y el número de épocas. Esta librería facilita la reproducibilidad de los experimentos al permitir ajustes sin modificar directamente el código fuente.

3.6 Protocolo experimental

Los experimentos realizados en esta tesis se estructuraron en tres etapas principales: pre-entrenamiento, poda y re-entrenamiento. Este protocolo experimental fue diseñado para evaluar el impacto de diferentes criterios de poda en modelos previamente seleccionados y entrenados, ajustando los niveles de compresión para alcanzar distintos factores de aceleración.

- Pre-entrenamiento

Aunque esta etapa ya ha sido descrita en detalle en secciones anteriores, cabe mencionar que los modelos iniciales utilizados en los experimentos se obtuvieron mediante un pre-entrenamiento en los conjuntos de datos seleccionados CIFAR-10, CIFAR-100 e ImageNet-1K. Estos modelos pre-entrenados son el punto de partida para la aplicación de las técnicas de poda.

- Poda

La poda se realiza de forma progresiva, utilizando un enfoque de poda global restringida. A diferencia de la poda local, que aplica una proporción fija de eliminación de parámetros en cada capa, la poda global elimina estructuras de toda la red hasta alcanzar un factor de aceleración predeterminado. Sin embargo, para evitar que grupos enteros de parámetros sean eliminados, lo que podría dañar severamente la funcionalidad del modelo, se preservó al menos un 10% de los parámetros dentro de cada grupo [49].

Para gestionar la poda estructurada, se utiliza el grafo de dependencias (DepGraph), el cual agrupa parámetros interdependientes en estructuras. Estos grupos fueron podados de acuerdo a un criterio de importancia seleccionado, que determina si un grupo resulta relevante al desempeño del modelo. Este enfoque asegura que la poda mantenga la consistencia estructural del modelo y su implementación en hardware se vea beneficiada de la reducción de parámetros.

La poda se lleva a cabo en hasta 400 pasos progresivos. Este número de pasos fue seleccionado para ofrecer un control fino sobre la reducción de FLOPs, evitando así una eliminación excesiva de parámetros y minimizando la pérdida de rendimiento. Cada paso implica la evaluación del modelo y la aplicación incremental de la poda hasta alcanzar el factor de aceleración deseado.

- Re-entrenamiento

Tras la aplicación de la poda, los modelos fueron re-entrenados para recuperar el desempeño perdido debido a la eliminación de parámetros. El programa de re-entrenamiento se diseñó para evitar el sobreajuste en los modelos.

El reentrenamiento se llevó a cabo durante un máximo de 100 épocas, utilizando una tasa de aprendizaje inicial de 0.01 que se redujo a una décima parte si la pérdida se estabilizaba durante 4 épocas consecutivas, implementando así una estrategia de reducción de *Plateau*. Además, se empleó *early stopping* con una paciencia de 10 épocas para interrumpir el entrenamiento si no se observa alguna disminución en la pérdida de la función de costo.

El optimizador utilizado fue Descenso de Gradiente Estocástico (SGD) con Nesterov Momentum configurado en 0.9, y el tamaño de lote se fijó en 128. La función de pérdida empleada fue la entropía cruzada (*Cross Entropy Loss*), la cual es adecuada para problemas de clasificación multiclase.

CAPÍTULO IV. COMPARACIÓN DE MÉTODOS DE PODA DE REDES NEURONALES

En este capítulo, se analizarán diversos métodos de poda clasificados según el criterio empleado para determinar la importancia de los filtros o estructuras en la red. Basándose en un *survey* reciente [50], los métodos de poda pueden agruparse en cuatro categorías principales:

1. Métodos dependientes de los pesos (*Weight-Dependent*): Estos métodos evalúan la importancia de los filtros considerando exclusivamente sus pesos, sin requerir datos de entrada.
2. Métodos basados en activaciones (*Activation-Based*): Determinan la relevancia de los filtros mediante el análisis de los mapas de activación generados por la red.
3. Métodos utilizando herramientas de optimización (*Optimization Tools*): Integran técnicas de optimización, como la expansión de Taylor, para identificar estructuras que pueden ser eliminadas.
4. Métodos con regularización (*Regularization*): Utilizan regularizadores de dispersión durante el entrenamiento para inducir la formación de estructuras dispersas en la red.

Cada uno de estos enfoques se describirá detalladamente en las secciones siguientes, haciendo énfasis en los aspectos matemáticos que sustentan su funcionamiento. Además, se analizarán sus ventajas y limitaciones, así como su impacto en el desempeño y la eficiencia de los modelos comprimidos.

Finalmente, se presentará una comparación entre los métodos seleccionados, destacando sus diferencias en términos de aplicabilidad, costo computacional y efectividad. Este análisis sirve como base para identificar los elementos más relevantes de cada enfoque y guiará el desarrollo de la metodología propuesta en capítulos posteriores.

4.1 Métodos de poda dependientes de los pesos

- Poda basada en normas ℓ_1 y ℓ_2

Estos métodos eliminan filtros de la red en función de su magnitud [51], medida mediante las normas ℓ_1 o ℓ_2 , en (4.1) se muestra la forma de calcular la norma ℓ_p de un filtro, donde p es el

orden del filtro, N_l es el número de canales de entrada de la capa l y K_l es el tamaño del kernel de la capa l .

$$\|F_i^l\|_p = \left(\sum_{n=1}^{N_l} \sum_{k_1=1}^{K_l} \sum_{k_2=1}^{K_l} |F_i^l(n, k_1, k_2)|^p \right)^{\frac{1}{p}} \quad (4.1)$$

La idea principal es que los filtros con magnitudes menores tienen menos impacto en el desempeño de la red porque generan activaciones de menor intensidad, lo que sugiere que su contribución a las decisiones de la red es menor.

- Media geométrica

Este método aproxima la media geométrica al suponer que un filtro de cada capa se acerca lo suficiente a dicho valor [52]. De esta manera, se elige entre los filtros $F_{i'}^l$ aquel que minimiza la suma de distancias euclidianas a todos los demás (4.2).

$$F_{i^*}^l \in \operatorname{argmin}_{i^* \in [1, N_{l+1}]} \sum_{i' \in [1, N_{l+1}]} \|x - F_{i'}^l\|_2 \quad (4.2)$$

Este filtro representa la media geométrica de la capa l . Los filtros correlacionados aportan información redundante, por lo que aquellos cercanos a la media geométrica se eliminan para mejorar la eficiencia del modelo.

4.2 Métodos de poda basados en activaciones

- Poda basada en rango de mapas de activación

El método HRank propuesto por Lin et al. [53] introduce un criterio basado en el rango de los mapas de activación para evaluar la importancia de los filtros en redes neuronales convolucionales. A diferencia de los métodos que emplean directamente propiedades de los pesos o las activaciones individuales, HRank determina la relevancia mediante el análisis del rango promedio de los mapas de características generados por cada filtro.

El método se fundamenta en la observación empírica de que el rango promedio de los mapas de activación generados por un filtro específico se mantiene constante, independientemente de la cantidad de imágenes que procese la red neuronal. Por tanto, este rango constituye una medida robusta y eficiente para estimar la cantidad de información contenida en dichos mapas. Los mapas con rangos altos se consideran ricos en información y contribuyen significativamente al desempeño del modelo, mientras que aquellos con rangos bajos contienen menos información y pueden ser eliminados con un menor impacto sobre la precisión.

Para evaluar matemáticamente el rango de los mapas de activación, se utiliza SVD. Un mapa de características $M_k^{(l)}$, generado por el filtro $F_k^{(l)}$, se puede descomponer como sigue:

$$M_k^{(l)} = \sum_{i=1}^r \sigma_i \mathbf{u}_i \mathbf{v}_i^T \quad (4.3)$$

donde σ_i , $\mathbf{u}_i$ y $\mathbf{v}_i^T$ corresponden, respectivamente, a los valores singulares, vectores singulares izquierdos y derechos del mapa de activación, y r es el rango del mapa. El mapa se puede aproximar a un rango menor $r' < r$, perdiendo así cierta información que refleja precisamente la importancia del mapa original:

$$M_k^{(l)} = \sum_{i=1}^{r'} \sigma_i \mathbf{u}_i \mathbf{v}_i^T + \sum_{i=r'+1}^r \sigma_i \mathbf{u}_i \mathbf{v}_i^T \quad (4.4)$$

De esta manera, los filtros que generan mapas de características con rangos consistentemente bajos a través de diferentes imágenes de entrada son considerados menos importantes y, por ende, candidatos a ser eliminados durante la poda.

La principal ventaja de HRank radica en que, al ser independiente de variaciones puntuales en las imágenes de entrada, permite una estimación del rango usando un número relativamente reducido de imágenes. Adicionalmente, no requiere modificaciones en la función de pérdida ni restricciones adicionales durante el entrenamiento, lo que facilita su implementación.

4.3 Métodos de poda utilizando herramientas de optimización

- Aproximación de Taylor de primer orden

Este método de poda estima la importancia de cada grupo de parámetros mediante expansiones de Taylor de la función de error, evaluando cómo variaría el rendimiento si se eliminaran dichos parámetros [54].

Para evitar evaluar el error del modelo según el número de parámetros, se utiliza una aproximación de Taylor de primer orden de la función del error, con esto se calcula una métrica que indica cuánto aumentaría el error si el conjunto estructural se estableciera en cero. Así, en lugar de depender únicamente del valor absoluto de los pesos, la importancia se determina por el efecto estimado sobre la pérdida del modelo. La importancia con una aproximación de primer orden para un conjunto de parámetros $\mathbf{w}_S$ puede aproximarse como:

$$I^{(1)}(\mathbf{w}_S) \approx \left(\sum_{w_i \in \mathbf{w}_S} g_i w_i \right)^2 \quad (4.5)$$

Donde g_i son los gradientes evaluados en la solución actual (4.6). De esta manera, en la poda se eliminan aquellas estructuras con menor importancia aproximada.

$$g_i = \frac{\partial E}{\partial w_i} \quad (4.6)$$

4.4 Métodos de poda con regularización

- Regularización creciente

En lugar de centrarse en establecer un criterio de importancia para determinar que filtros eliminar, este método se centra en proponer un programa de poda a través de una etapa de regularización creciente [55].

Primero, se calcula la norma ℓ_1 de cada grupo de pesos y aquellos con menor norma ℓ_1 se consideran candidatos a ser eliminados, pero, a diferencia del procedimiento tradicional en el cual se podan los pesos elegidos inmediatamente, este método introduce una etapa intermedia de regularización creciente donde se aumenta gradualmente la penalización ℓ_2 sobre esos grupos para impulsar sus pesos hacia cero sin interrumpir bruscamente el entrenamiento, y finalmente los

poda. De esta manera, el modelo se adapta gradualmente a la pérdida de dichos filtros, minimizando el impacto en su rendimiento.

Una forma resumida de describir la función objetivo es:

$$L_{\text{total}}(w) = L_{\text{tarea}}(w) + \sum_{i:M_i=0} \lambda_i \|w_i\|_2^2, \quad \text{con } \lambda_i \leftarrow \lambda_i + \delta\lambda. \quad (4.7)$$

En esta ecuación, $L_{\text{tarea}}(w)$ es la pérdida principal, w_i es el conjunto de pesos del filtro a podar, M_i una máscara binaria que indica cuáles filtros serán removidos eventualmente, y λ_i la penalización ℓ_2 que se incrementa progresivamente.

4.5 Resultados experimentales y análisis comparativo

En esta sección se presentan los resultados obtenidos tras aplicar los métodos de poda descritos previamente en diversas arquitecturas de redes neuronales profundas. Se analiza cómo estos métodos afectan al desempeño de los modelos en términos de precisión y eficiencia computacional, considerando diferentes niveles de reducción de parámetros y operaciones. El objetivo de este análisis comparativo es proporcionar una perspectiva del impacto que tienen las diferentes estrategias de poda sobre los modelos evaluados, así como determinar las condiciones bajo las cuales estas técnicas resultan más efectivas.

4.5.1 Análisis de la relación entre exactitud y parámetros podados

En esta subsección se estudia cómo la exactitud (*accuracy*) original de las redes neuronales profundas se ve afectada inmediatamente después de aplicar los diferentes métodos de poda, sin haber realizado aún el proceso de re-entrenamiento. Las figuras que se presentan a continuación ilustran claramente la relación entre el porcentaje de parámetros eliminados y el *accuracy* resultante para las arquitecturas evaluadas (VGG19-BN, ResNet-18, ResNet-50, y DenseNet-121), entrenadas en los conjuntos de datos CIFAR-10 (líneas continuas) y CIFAR-100 (líneas punteadas). Las siguientes gráficas muestran estos resultados:

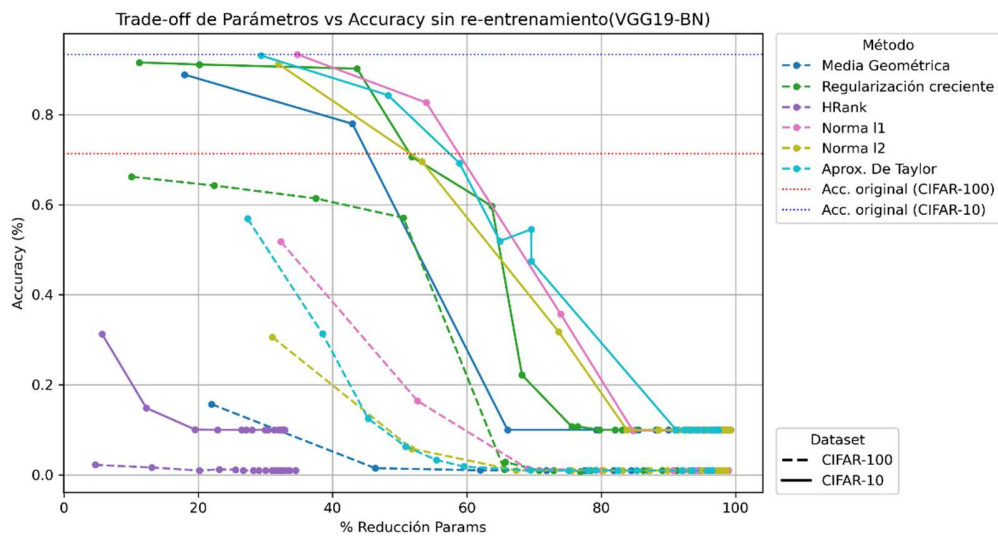


Figura 4.1. Relación entre porcentaje de parámetros podados y precisión sin re-entrenamiento para VGG19-BN.

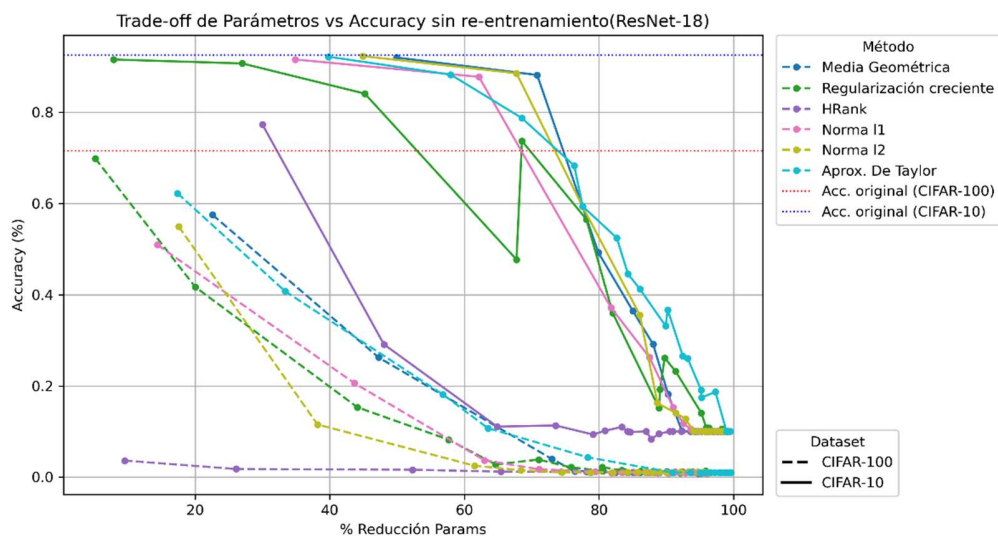


Figura 4.2. Relación entre porcentaje de parámetros podados y precisión sin re-entrenamiento para ResNet-18.

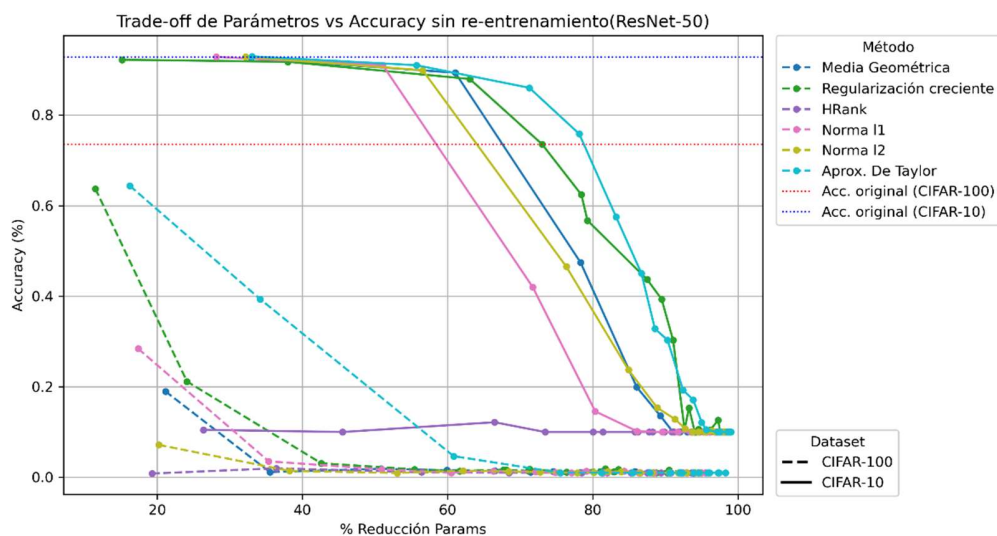


Figura 4.3. Relación entre porcentaje de parámetros podados y precisión sin re-entrenamiento para ResNet-50.

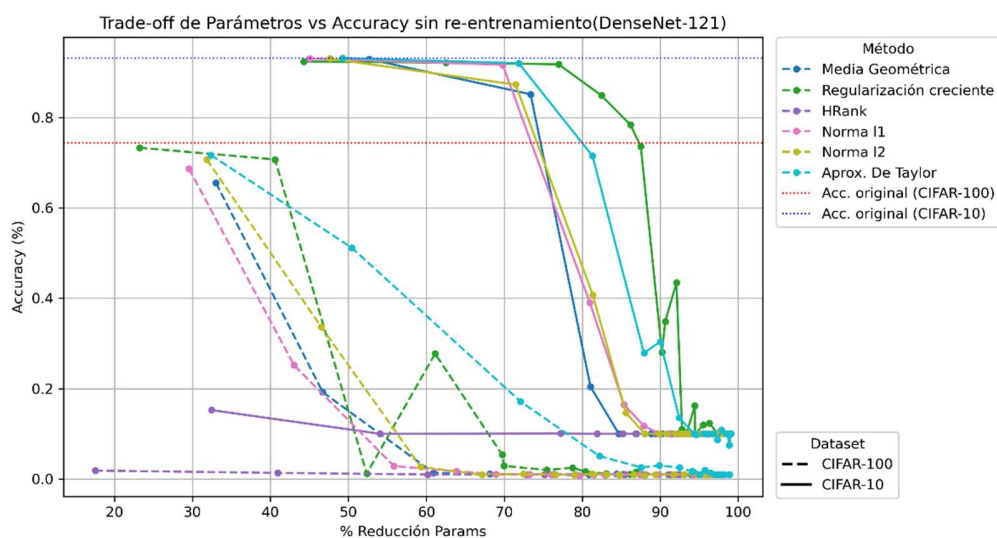


Figura 4.4. Relación entre porcentaje de parámetros podados y precisión sin re-entrenamiento para DenseNet-121.

A partir de las gráficas mostradas en las Figuras 4.1-4.4 se puede observar que el desempeño original de una red neuronal profunda se ve directamente afectado por el proceso de poda. Conforme incrementa el porcentaje de parámetros eliminados, la precisión del modelo disminuye progresivamente, mostrando una clara relación negativa entre la magnitud de la poda y el desempeño inmediato del modelo. Es importante notar que esta disminución de la precisión se

presenta en todos los métodos evaluados, lo que indica que ningún método de poda genera una mejora en el desempeño original sin un proceso adicional de re-entrenamiento.

Así, se observa que el proceso de re-entrenamiento es esencial y fundamental para recuperar el desempeño perdido tras la aplicación de cualquier técnica de poda. Sin esta etapa posterior de ajuste de parámetros, los modelos podados no solo pierden significativamente precisión, sino que tampoco muestran signos de recuperación espontánea o estabilización.

Finalmente, destaca la sensibilidad diferenciada que presentan los modelos según la complejidad del conjunto de datos con el que fueron entrenados. Particularmente se observa, que los modelos entrenados con CIFAR-100 sufren una degradación de exactitud más rápida y severa al eliminar una proporción menor de parámetros, en comparación con aquellos entrenados con CIFAR-10. Este fenómeno puede atribuirse a que los modelos entrenados sobre bases de datos más complejas tienden a formar representaciones internas más especializadas, haciéndolos menos robustos frente a cambios estructurales abruptos generados por la poda. Por lo tanto, se recomienda aplicar estrategias de poda más cuidadosas y moderadas cuando se trabaja con tareas y conjuntos de datos más complejos.

4.5.2 Impacto del re-entrenamiento en la exactitud vs reducción de FLOPs

En esta subsección se analiza la recuperación del desempeño de los modelos tras realizar el proceso de re-entrenamiento posterior a la poda, haciendo énfasis en la reducción de las FLOPs. Las gráficas presentadas a continuación muestran la relación entre el porcentaje de reducción en FLOPs y la precisión alcanzada tras el re-entrenamiento para las cuatro arquitecturas estudiadas.

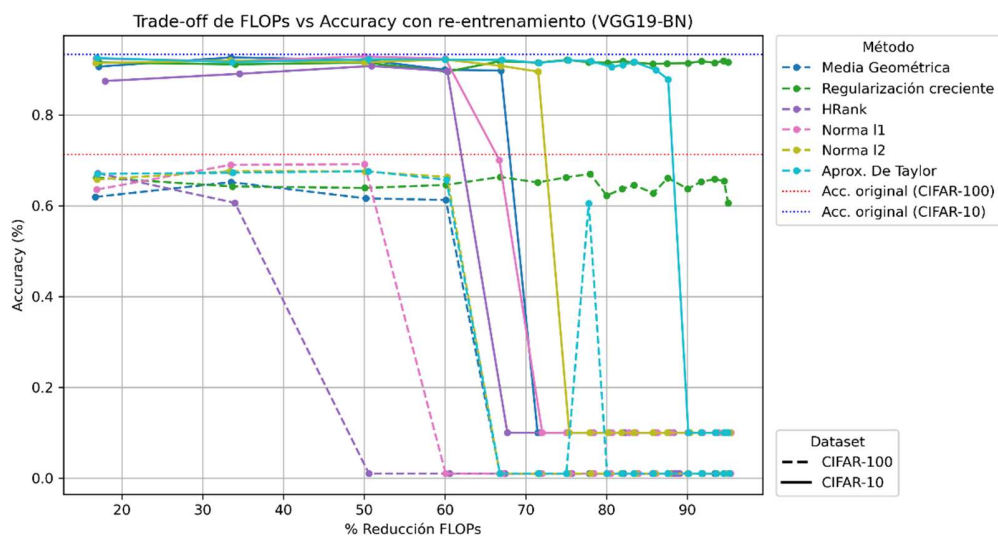


Figura 4.5. Relación entre reducción de FLOPs y precisión con re-entrenamiento para VGG19-BN.

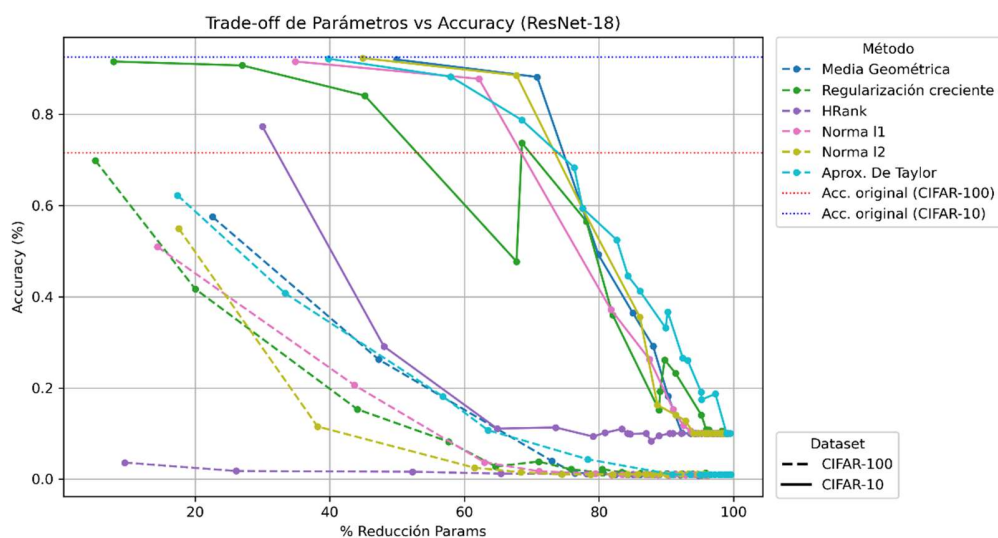


Figura 4.6. Relación entre reducción de FLOPs y precisión con re-entrenamiento para ResNet-18.

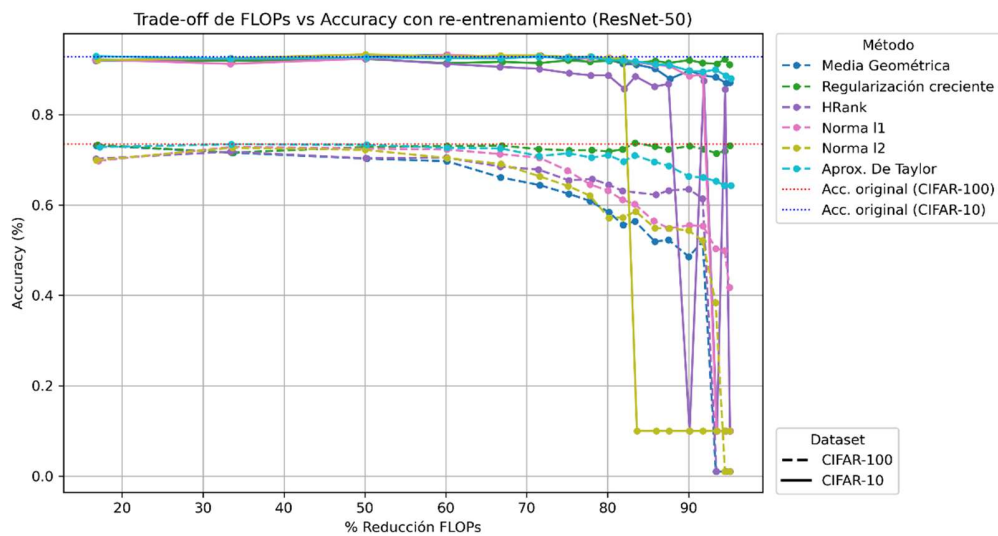


Figura 4.7. Relación entre reducción de FLOPs y precisión con re-entrenamiento para ResNet-50.

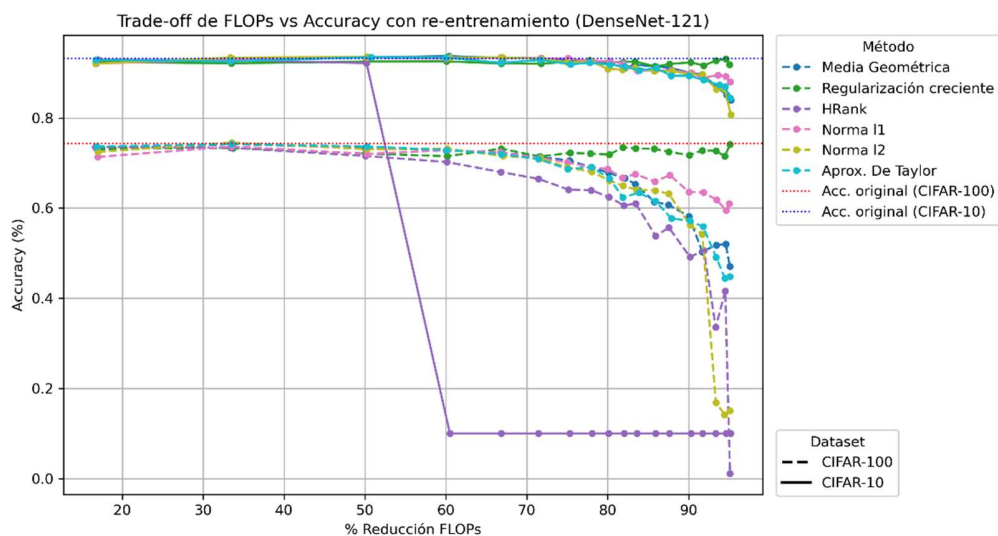


Figura 4.8. Relación entre reducción de FLOPs y precisión con re-entrenamiento para DenseNet-121.

La reducción de FLOPs suele ser más relevante que la reducción de parámetros, ya que una disminución significativa en FLOPs está directamente asociado a un menor tiempo de inferencia y menor consumo energético, factores cruciales especialmente en entornos con recursos limitados.

Observando los resultados experimentales de las Figura 4.7 y Figura 4.8, todos los métodos de poda permiten lograr al menos una reducción de FLOPs del 50%, conservando el desempeño con

pérdidas mínimas en *accuracy* en ambas bases de datos CIFAR-10 y CIFAR-100 en los modelos de ResNet-50 y DenseNet-121. Estos resultados sugieren una capacidad excesiva de representación en las redes originales con relación al nivel de complejidad de los conjuntos de datos utilizados, indicando redundancia en sus estructuras.

Particularmente, el método de regularización creciente destaca al inducir estructuras dispersas antes de realizar la poda, permitiendo lograr altas tasas de aceleración con mínima pérdida de desempeño. En segundo lugar, el método basado en aproximaciones de Taylor también demuestra buenos resultados. Cabe resaltar que para ciertas redes como VGG19-BN y ResNet-50, reducir excesivamente los FLOPs genera una caída abrupta en la capacidad de clasificación, marcando un límite, en el caso de estas arquitecturas, en el grado de compresión posible.

Finalmente, se aprecia una caída más temprana y pronunciada en los modelos entrenados con CIFAR-100 respecto a CIFAR-10, reafirmando la sensibilidad de las redes a la poda frente a tareas más complejas.

4.5.3 Análisis de la eficiencia del re-entrenamiento tras poda

En esta subsección se analiza la relación entre el porcentaje de parámetros eliminados y el número de épocas necesarias para recuperar el desempeño original del modelo tras la poda, mediante el re-entrenamiento. Cabe resaltar que no en todos los casos se recupera el desempeño original. En las siguientes figuras, se muestran las gráficas que auxilian este análisis.

De los resultados obtenidos se observa que existe un rango específico de reducción de parámetros donde las redes neuronales logran recuperar su desempeño original sin un incremento significativo en el número de épocas requeridas para el re-entrenamiento. Por ejemplo, en la Figura 4.10 se aprecia que el entrenamiento de los modelos a los que se les redujo menos del 80% de sus parámetros constó de entre 25 y 50 épocas. Un comportamiento similar se observa en la Figura 4.12, donde el entrenamiento de los modelos a los que se podó entre un 0% y 85% de sus parámetros no supera las 50 épocas y una vez superado este rango se dispara el número de épocas de entrenamiento necesarias .

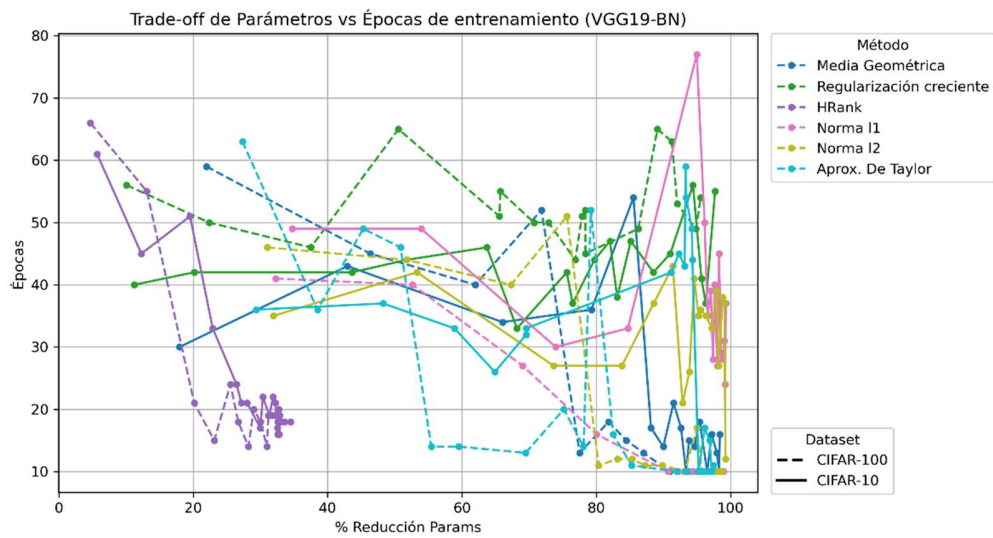


Figura 4.9. Relación entre reducción de parámetros y número de épocas de re-entrenamiento para VGG19-BN.

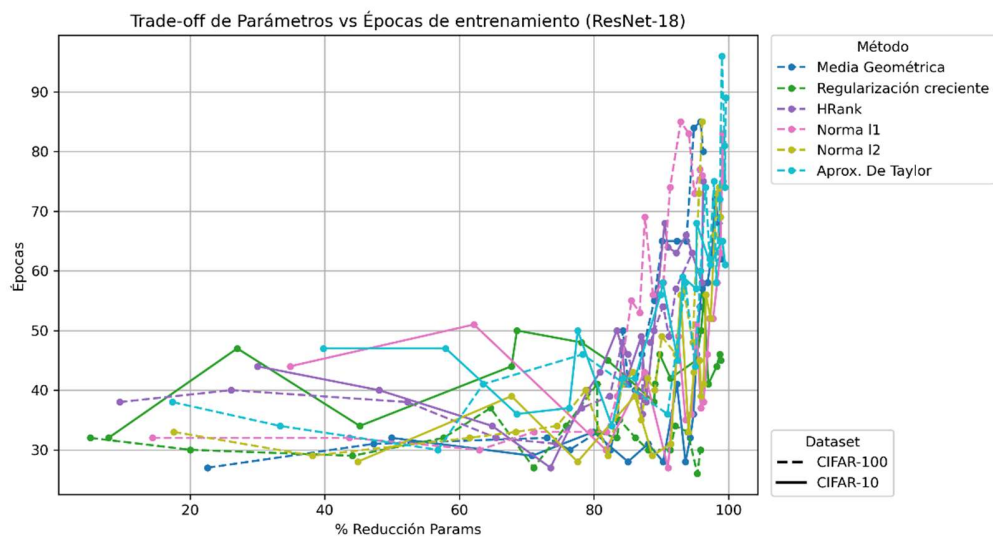


Figura 4.10. Relación entre reducción de parámetros y número de épocas de re-entrenamiento para ResNet-18.

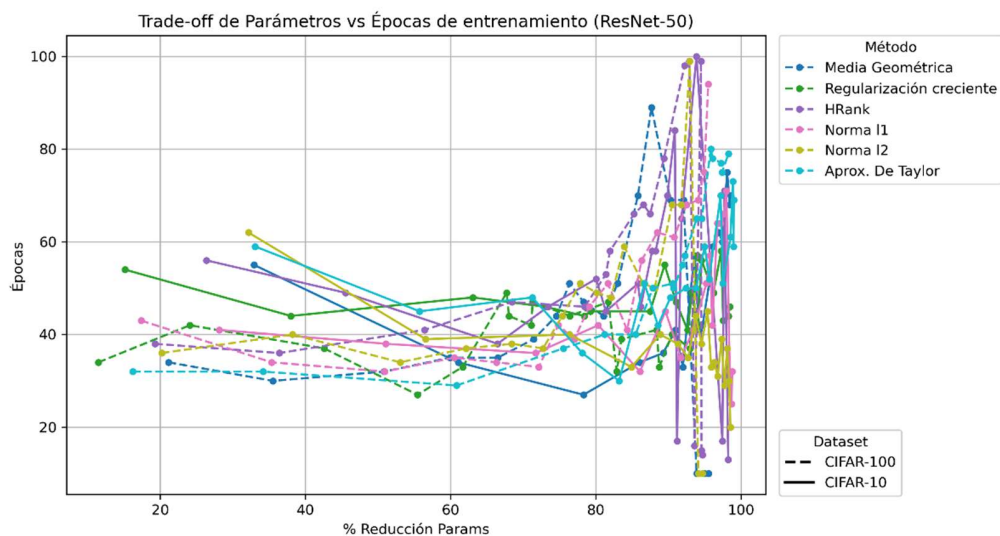


Figura 4.11. Relación entre reducción de parámetros y número de épocas de re-entrenamiento para ResNet-50.

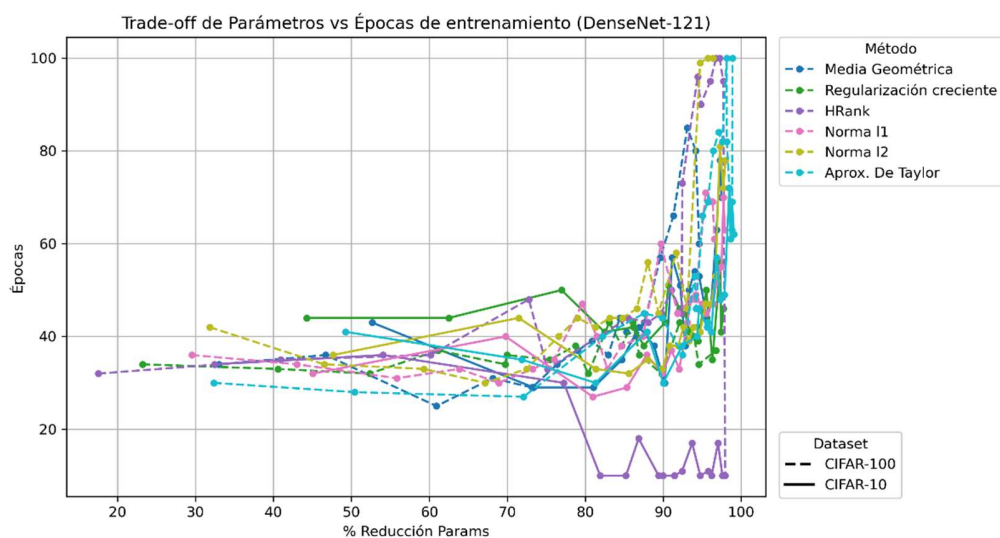


Figura 4.12. Relación entre reducción de parámetros y número de épocas de re-entrenamiento para DenseNet-121.

Esto implica que, dentro de dicho rango, la poda no afecta de manera considerable la capacidad de aprendizaje y generalización del modelo, lo que a su vez indica que existe un punto de compresión que equilibra la reducción de complejidad y la eficiencia del re-entrenamiento.

Sin embargo, al sobrepasar este rango, se hace evidente que la dificultad de recuperar el desempeño del modelo se incrementa notablemente, reflejada en un aumento considerable del

número de épocas necesarias para el re-entrenamiento. Este fenómeno puede interpretarse como una pérdida importante en la capacidad de representación del modelo, dificultando el proceso de recuperación del desempeño.

4.5.4 Modelos que mejoran el accuracy original tras poda y re-entrenamiento

Esta subsección presenta un análisis detallado de aquellos casos en los cuales el proceso combinado de poda y re-entrenamiento no solo recupera el desempeño original del modelo, sino que incluso logra mejorarlo. En la Tabla 4.1 se resumen estos casos para los conjuntos CIFAR-10 y CIFAR-100.

Tabla 4.1. Modelos entrenados que superan el *accuracy* original.

Arquitectura	Conjunto de datos	Factor de aceleración alcanzado	Método de poda	% Red. de Parám.	% Red. de FLOPs	Acc inicial	Accuracy luego de la poda y re-entrenamiento	Δ Acc
ResNet-50	CIFAR-10	1.2	Aprox. De Taylor	33.08%	16.77%	92.78%	92.94%	0.16%
		3.5		86.71%	71.49%		92.83%	0.05%
		4.5		90.31%	77.99%		92.80%	0.02%
		2.5	Media Geométrica	86.06%	60.13%		93.27%	0.49%
		3.5		91.00%	71.72%		93.03%	0.25%
		2.5	Norma l1	80.35%	60.23%		93.25%	0.47%
		3		86.09%	66.78%		92.84%	0.06%
		3.5		89.61%	71.56%		93.10%	0.32%
		2	Norma l2	76.36%	50.07%		93.34%	0.56%
		2.5		84.92%	60.18%		92.87%	0.09%
		3		88.87%	66.79%		93.10%	0.32%
		3.5		91.30%	71.54%		93.11%	0.33%
		4		92.73%	75.25%		92.81%	0.03%
		4.5		93.79%	77.86%		92.89%	0.11%
	CIFAR-100	1.5	EigenCAM	51.02%	33.39%		93.24%	0.46%
		2.5		82.06%	60.08%		92.83%	0.05%
DenseNet-121	CIFAR-10	6	Regularización creciente	81.73%	83.38%	93.17%	73.75%	0.23%
		2	Aprox. De Taylor	81.32%	50.77%		93.51%	0.34%
		2.5		87.92%	60.02%		93.39%	0.22%
		1.5	Media Geométrica	73.38%	33.53%		93.18%	0.01%
		2		81.06%	50.37%		93.47%	0.30%
		2.5		84.69%	60.39%		93.72%	0.55%
		3		86.94%	67.04%		93.28%	0.11%
		3.5		88.83%	71.82%		93.27%	0.10%
		2	Norma l1	80.97%	50.23%		93.40%	0.23%
		3		87.89%	66.79%		93.38%	0.21%
		3.5		89.91%	71.71%		93.23%	0.06%
		4		91.08%	75.11%		93.22%	0.05%
	CIFAR-100	1.5	Norma l2	71.46%	33.36%		93.38%	0.21%
		2		81.36%	50.25%		93.55%	0.38%
		2.5		85.61%	60.49%		93.40%	0.23%
		3		88.02%	66.92%		93.45%	0.28%

De los resultados se observa que, para CIFAR-10, diversos métodos de poda lograron superar la exactitud original tras el re-entrenamiento, lo cual indica que una reducción moderada y adecuada de parámetros puede conducir a un modelo más generalizable y eficaz. Destacan especialmente las arquitecturas DenseNet-121 y ResNet-50, las cuales mostraron mayor resiliencia a la poda, probablemente debido a sus conexiones residuales y de salto que les permiten mantener una buena representación frente a la poda estructural.

Por otro lado, con CIFAR-100, solamente 2 métodos lograron mejorar el *accuracy* original. Este resultado refleja la mayor complejidad del conjunto de datos CIFAR-100 y subraya la sensibilidad del modelo frente a la eliminación excesiva de parámetros cuando la tarea es más compleja. Estos hallazgos sugieren que, mientras una poda efectiva puede mejorar la capacidad general del modelo en tareas relativamente simples, se debe ejercer mayor precaución y moderación al aplicar poda en escenarios con bases de datos más complejas.

4.6 Discusión general

A partir del análisis realizado en las secciones anteriores, se pueden extraer conclusiones generales acerca del comportamiento de los métodos de poda evaluados y se obtienen recomendaciones para su aplicación en redes neuronales profundas.

En primer lugar, queda demostrado que cualquier método de poda requiere necesariamente un proceso de re-entrenamiento posterior para recuperar, y en ciertos casos, incluso superar el desempeño original del modelo. La poda, al eliminar estructuras redundantes o menos relevantes, afecta la precisión del modelo de manera significativa si no se complementa con un ajuste posterior que permita la adaptación a las modificaciones realizadas en su estructura interna.

La comparación de métodos de poda muestra también diferencias considerables entre los métodos analizados. Entre ellos, el método basado en regularización creciente destaca consistentemente, mostrando capacidad para inducir estructuras esparcidas y alcanzando mayores niveles de aceleración con pérdidas mínimas de desempeño. El método basado en aproximaciones de Taylor también sobresale, al proporcionar criterios más informativos sobre la importancia de las estructuras en términos de su contribución al error del modelo.

Por otro lado, los resultados experimentales indican que no todas las arquitecturas reaccionan de igual manera ante la poda. Arquitecturas como DenseNet-121 y ResNet-50 mostraron mayor resiliencia y capacidad de recuperación, posiblemente debido a sus conexiones residuales y densas, las cuales facilita el flujo de gradientes y permite retener una representación del conjunto de datos tras eliminar estructuras.

La complejidad del conjunto de datos también desempeña un papel fundamental en la efectividad de la poda. Modelos entrenados en conjuntos de datos más complejos, como CIFAR-100, experimentan caídas más rápidas en desempeño al incrementar la magnitud de la poda, lo que implica que para tareas más difíciles es indispensable aplicar estrategias de poda más moderadas y cuidadosamente controladas.

Asimismo, se identificó que puede existir punto óptimo en la magnitud de la poda, donde la reducción de parámetros y FLOPs proporciona beneficios en términos de eficiencia computacional sin tener que pagar altos costos adicionales de re-entrenamiento. Superar este umbral provoca un aumento considerable del esfuerzo necesario para restaurar o mejorar el desempeño original del modelo, lo que limita la ventaja práctica de una poda agresiva.

En términos generales, para aplicar métodos de poda de manera efectiva se recomienda:

- Seleccionar métodos de poda que incorpore criterios informativos sobre la relevancia estructural de los parámetros, pero además toman en cuenta el programa de poda, tales como regularización creciente.
- Considerar las características específicas de la arquitectura seleccionada, dando prioridad a aquellas que cuenta con conexiones residuales o densas para aplicaciones donde se anticipan niveles significativos de compresión.
- Adecuar el nivel de poda en función de la complejidad de la tarea y del conjunto de datos, realizando pruebas preliminares que permitan identificar el punto óptimo de compresión.
- Asegurar un proceso de re-entrenamiento suficientemente robusto, con estrategias de reducción de tasa de aprendizaje y *early stopping*, para maximizar la recuperación del desempeño original tras la poda.

CAPÍTULO V. EVALUACIÓN DE TÉCNICAS DE XAI PARA LA PODA ESTRUCTURAL

Una de sus principales deficiencias de las DNNs radica en su complejidad y en la falta de interpretabilidad que esta conlleva, haciendo que estos modelos funcionen como cajas negras, donde el usuario se encuentra en la necesidad de confiar en los resultados obtenidos por el modelo sin ningún tipo de explicación que los justifique.

Para esto, las técnicas de XAI ha surgido como respuesta a este problema. A través de información adicional a las inferencias que realiza un modelo de DNN, se busca proporcionar explicaciones claras y entendibles sobre cómo los modelos complejos toman decisiones o realizan predicciones.

Las técnicas de XAI permite identificar qué partes del modelo influyen más significativamente en sus resultados y cuáles podrían considerarse prescindibles o redundantes. Así, el empleo de métodos de explicabilidad tiene el potencial de aportar criterios más informados y robustos para seleccionar elementos específicos que se pueden eliminar, manteniendo al mismo tiempo el desempeño del modelo. Este capítulo explora las técnicas de XAI relevantes para la presente tesis, detallando su fundamentación, aplicación, y la manera en que pueden ser integradas en un método de poda.

5.1 Métodos basados en CAM

Los métodos basados en mapas de activación de clase (*Class Activation Mapping*, CAM) fueron elegidos como auxiliares de la poda estructural debido a su capacidad para proporcionar explicaciones visuales claras e intuitivas sobre las decisiones tomadas por modelos de redes neuronales profundas en tareas de clasificación visual.

Dentro de los métodos basados en CAMs se puede distinguir entre métodos que utilizan relevancia mediante gradientes y métodos que emplean perturbaciones para generar los mapas visuales que resaltan las regiones de una imagen más influyentes en la decisión del modelo. Entre los métodos basados en gradientes destacan Grad-CAM [56] y Grad-CAM++ [57], mientras que Score-CAM [58] y Ablation-CAM [59] son ejemplos de métodos basados en perturbaciones. Estos métodos permiten una interpretación visual intuitiva y una buena capacidad para adaptarse a

diversas arquitecturas neuronales. Además, permiten identificar directamente regiones críticas en las imágenes, lo que es particularmente útil en tareas de visión por computadora

Estos métodos destacan por ser computacionalmente eficientes, especialmente los basados en gradientes, lo cual es esencial para la implementación práctica en escenarios de poda estructural.

5.1.1 Class Activation Mapping

El método original de CAM [60] fue propuesto para generar mapas visuales que indican las regiones más relevantes de una imagen para la clasificación realizada por una CNN. Matemáticamente, un CAM se calcula ponderando los mapas de activación generados por la última capa convolucional del modelo mediante pesos derivados directamente de la capa totalmente conectada, la cual sigue a una capa de *Global Average Pooling* (GAP).

La construcción del CAM se basa en que cada filtro de la última capa convolucional puede interpretarse como un detector de patrones específicos. Al aplicar una capa GAP sobre esta última capa convolucional, se obtiene un valor promedio para cada mapa de activación, resultando en una única característica representativa por mapa. Estos valores promedio se alimentan directamente a una capa totalmente conectada que calcula la puntuación de clasificación para cada clase. Los pesos que conectan estos valores promediados con la salida del clasificador pueden verse como una medida directa de la importancia relativa de cada mapa de activación para la clase específica. Por lo tanto, al multiplicar cada mapa de activación por sus respectivos pesos y sumarlos, se obtiene un mapa que indica las regiones más relevantes en la imagen original para la clasificación hecha por el modelo. En la Figura 5.1 se ilustra el proceso de obtención de un CAM a partir de una CNN.

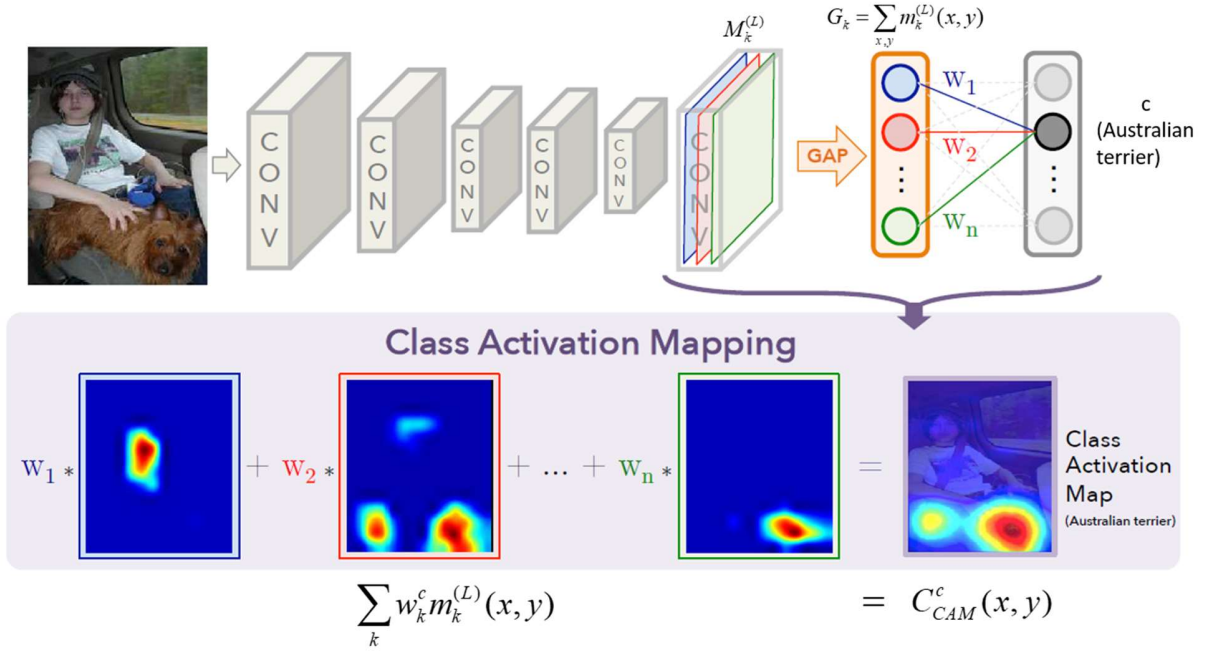


Figura 5.1. Diagrama de obtención de CAM [60].

La formulación general del CAM es:

$$C_{CAM}^c(x,y) = \sum_k w_k^c m_k^{(L)}(x,y) \quad (5.1)$$

donde $m_k^{(L)}(x,y)$ representa la activación del k -ésimo mapa de activación en la posición espacial (x,y) de la capa L , y w_k^c es el peso asociado al k -ésimo filtro obtenido directamente de la capa totalmente conectada después de una capa de GAP correspondiente a la clase c .

5.1.2 Gradient-weighted Class Activation Mapping

Grad-CAM (*Gradient-weighted Class Activation Mapping*) fue introducido por Selvaraju et al. para superar las limitaciones del método CAM original [61], especialmente la restricción de que la arquitectura del modelo debe incluir obligatoriamente una capa GAP seguida de una capa totalmente conectada. Esto restringía notablemente la aplicabilidad de CAM, excluyendo así muchas arquitecturas modernas que no siguen esta estructura específica.

Grad-CAM resuelve esta limitación utilizando los gradientes de la puntuación de la clase objetivo con respecto a los mapas de activación generados por la última capa convolucional. La

intuición detrás de este enfoque es que los gradientes indican la sensibilidad del puntaje de clasificación para una clase específica respecto a cada activación de la última capa. Estos gradientes, por lo tanto, pueden interpretarse como indicadores directos de la relevancia de cada región en los mapas de activación.

La Figura 5.2 ilustra el funcionamiento del método Grad-CAM aplicado a una CNN. En este esquema, una imagen de entrada es procesada por la CNN, produciendo un conjunto de mapas de características en la última capa convolucional, denotados como $M_k^{(L)}$. Estos mapas capturan la información espacial y semántica relevante detectada por los diferentes filtros de la red. La salida de la red avanza hacia las capas totalmente conectadas, que generan las puntuaciones de clasificación y para cada una de las clases posibles.

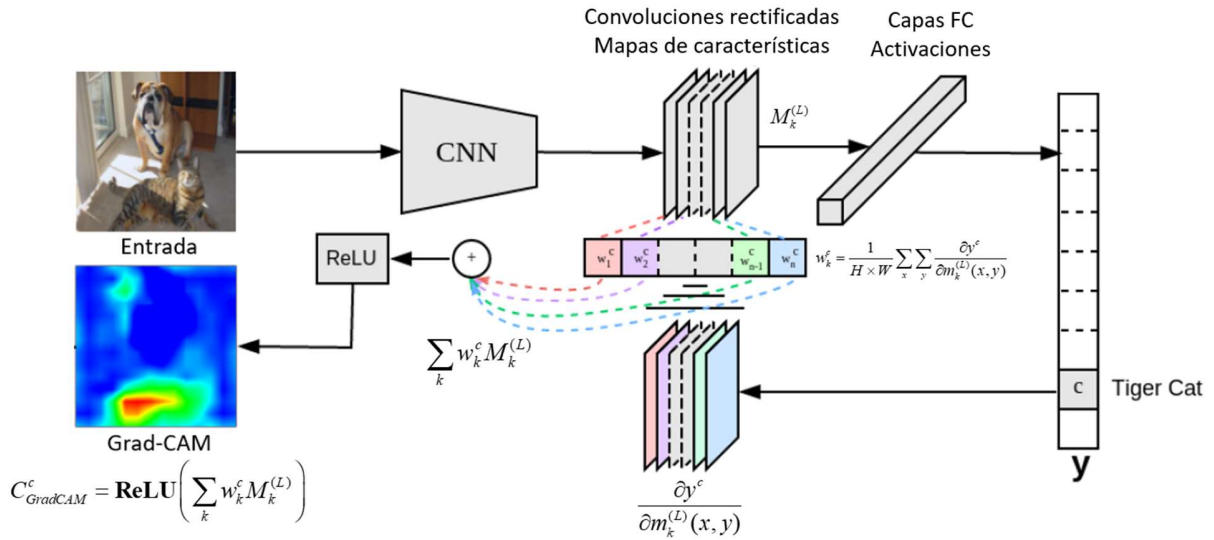


Figura 5.2. Diagrama de obtención de Grad-CAM [61].

Para calcular el Grad-CAM correspondiente a una clase objetivo c , primero se computan los gradientes del puntaje de la clase y^c respecto a cada posición espacial de los mapas de activación $M_k^{(L)}$. El promedio global de estos gradientes en cada mapa (Ecuación (5.2)) determina los pesos w_k^c , que cuantifican la relevancia de cada mapa de características para la clase c .

$$w_k^c = \frac{1}{H \times W} \sum_i \sum_j \frac{\partial y^c}{\partial m_k^{(L)}(i, j)} \quad (5.2)$$

Aquí, y^c representa la puntuación de salida del modelo para la clase c , mientras que H y W son las dimensiones de largo y ancho del mapa de activación. Esta formulación permite que Grad-CAM sea aplicable directamente a cualquier arquitectura sin modificaciones adicionales, esto facilita su uso generalizado en comparación con el método original de CAM.

Posteriormente, cada mapa de características se multiplica por su peso correspondiente y se suman todos los mapas ponderados. Finalmente, se aplica una función de activación ReLU para obtener el mapa de activación de clase $C_{GradCAM}^c$, que resalta las regiones de la imagen que más contribuyen a la predicción de la clase objetivo.

Matemáticamente, Grad-CAM se define como:

$$C_{GradCAM}^c = \text{ReLU} \left(\sum_k w_k^c M^k \right) \quad (5.3)$$

5.1.3 Variantes recientes de CAM

Además de los métodos pioneros como CAM y Grad-CAM, recientemente han surgido numerosas variantes que buscan mejorar la precisión y utilidad de los mapas de relevancia generados. Estas variantes adaptan la idea original de CAM utilizando diferentes estrategias para ponderar las activaciones convolucionales y gradientes, con el objetivo de obtener representaciones más fieles y útiles para tareas específicas, como la poda estructural de redes neuronales.

HiResCAM [62] propone una variante que realiza multiplicaciones elemento a elemento entre activaciones y gradientes antes de combinarlos globalmente. Esta aproximación mantiene detalles más finos y resolución espacial superior a Grad-CAM, lo que podría proporcionar información más precisa para la selección de filtros relevantes.

Grad-CAM++ [63] extiende la técnica original de Grad-CAM al incluir términos de segundo orden del gradiente. Esto mejora la representación de objetos repetidos o múltiples instancias del objeto objetivo, brindando mapas más informativos y capaces de capturar dependencias espaciales complejas.

XGrad-CAM [64] aplica un escalado dinámico a los gradientes usando las activaciones normalizadas antes de agregarlas globalmente. Este ajuste dinámico enfatiza o atenúa gradientes según la intensidad de la respuesta del filtro, facilitando mapas de relevancia más estables y consistentes.

AblationCAM [65] evalúa directamente la importancia de cada filtro mediante la técnica de ablación, estableciendo activaciones en cero y midiendo el impacto en la probabilidad de la clase objetivo. Este método es particularmente valioso para interpretaciones cuantitativas robustas.

ScoreCAM [66] prescinde completamente del uso directo de gradientes. En cambio, genera mapas de relevancia perturbando la imagen original mediante activaciones escaladas y mide cómo estas perturbaciones afectan la salida del modelo. Esto ofrece una aproximación sin ruido a los mapas de relevancia, eliminando fluctuaciones gradientes.

EigenCAM [67] simplifica el proceso utilizando análisis por componentes principales (PCA) en el tensor de activaciones, sin discriminación explícita por clase. La proyección en el primer componente principal destaca regiones energéticamente relevantes, siendo especialmente útil para visualizaciones generales y robustas.

EigenGradCAM [67] combina EigenCAM y discriminación de clases aplicando PCA sobre el producto activación-gradiente. Esto genera mapas similares a Grad-CAM, pero notablemente más limpios y enfocados.

LayerCAM [68] pondera espacialmente las activaciones con los gradientes positivos dentro de la misma capa, generando mapas altamente detallados y específicos por capa. Es particularmente útil para identificar características relevantes en etapas tempranas del modelo.

FullGrad [69] aprovecha el aprendizaje de toda la red, sumando los gradientes provenientes de los sesgos de múltiples capas, proporcionando así una visión completa de cómo cada parte del modelo contribuye a la decisión final.

KPCA-CAM [70] emplea Kernel PCA sobre las activaciones para capturar relaciones no lineales complejas que podrían pasar desapercibidas con PCA estándar. Este enfoque es valioso cuando la relación entre características y decisión del modelo es intrínsecamente no lineal.

FEM (*Feature Explanation Method*) [71] utiliza un enfoque libre de gradientes que binariza cada activación según un umbral adaptativo, integrando únicamente aquellas activaciones más significativas en el mapa final. Este método es ideal para situaciones donde se requieren decisiones binarias claras sobre relevancia.

FinerCAM [72] busca optimizar la clasificación de clases muy similares. Lo hace contrastando activamente pares de clases cercanas, suprimiendo rasgos compartidos y destacando las características más discriminativas, generando así mapas precisos y específicos.

La ponderación obtenida mediante métodos CAM refleja la relevancia de cada filtro respecto a las decisiones del modelo, lo que hace posible utilizar esta información como criterio para la poda estructural. Aquellos filtros asociados con ponderaciones bajas serán considerados menos relevantes y, por lo tanto, candidatos para ser eliminados, permitiendo una poda informada y orientada a mantener el rendimiento predictivo del modelo. Estas variantes ofrecen enfoques diversos para generar mapas de relevancia, a continuación, se evalúan y comparan para determinar cuál es más útil según el objetivo de la poda estructural de redes neuronales profundas.

5.2 Evaluación comparativa de métodos basados en CAM

El objetivo principal de esta sección es realizar una evaluación comparativa entre 13 métodos basados en CAM, con la finalidad de identificar cuáles asignan de manera más efectiva la importancia a los filtros generados por las capas convolucionales en redes neuronales profundas. La correcta asignación de importancia permite utilizar estos valores como criterio clave para realizar poda estructural, eliminando aquellos filtros con menor contribución a la precisión del modelo. Esta evaluación se realizó desde dos enfoques complementarios: uno cualitativo y otro cuantitativo, a fin de obtener una perspectiva sobre el desempeño de cada método.

5.2.1 Evaluación cualitativa

La evaluación cualitativa se fundamenta en la inspección visual de los mapas generados por cada método CAM. Esta inspección se hace con la intención de verificar intuitivamente si los mapas destacan con precisión y claridad el objeto de interés en las imágenes evaluadas, lo que sugiere una mejor asignación de relevancia a las regiones activadas por los filtros involucrados en la predicción.

Para realizar esta evaluación se seleccionaron 10 imágenes provenientes del conjunto de datos ImageNet-1k. Para cada imagen, se generaron CAMs en cuatro etapas diferentes de la arquitectura ResNet-50, tal como se presenta en la Figura 5.3. Esta decisión se tomó considerando que la poda estructural debe aplicarse en distintas profundidades del modelo, y no únicamente en la última capa convolucional, como suele hacerse en los métodos basados en CAM.

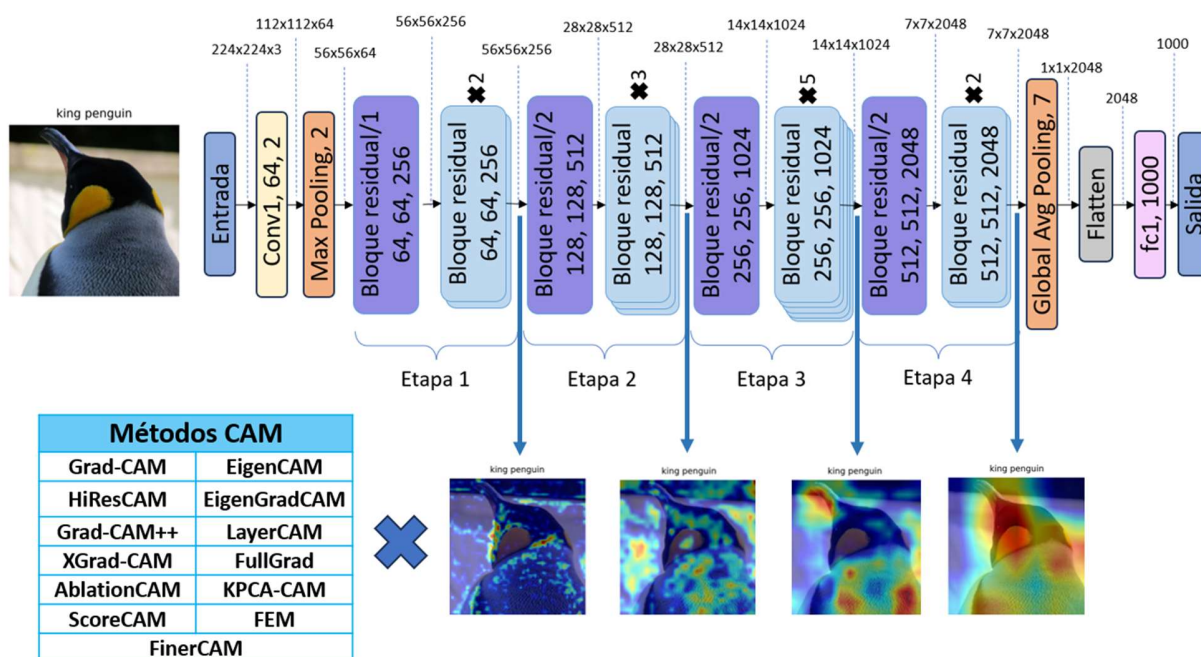


Figura 5.3. Generación de CAMs de ResNet-50.

En total, se evaluaron visualmente 40 mapas por cada método. Se observó un comportamiento general diferenciado entre etapas. Los CAMs generados a partir de las activaciones de salida de la etapa 1 se muestran en la Figura 5.4. En las primeras etapas, 1 y 2, los mapas generados suelen ser difusos, mostrando puntos de calor muy aislados o activaciones amplias que no se centran claramente en regiones específicas. La mayoría de los métodos presentan activaciones en el fondo o zonas irrelevantes del objeto, con poca coherencia semántica. Tal es el caso de Grad-CAM++,

XGrad-CAM, EigenGrad-CAM, entre otros. Algunos métodos tienden a responder principalmente a patrones básicos de bajo nivel, como bordes, texturas y colores, tales como EigenCAM y ScoreCAM.

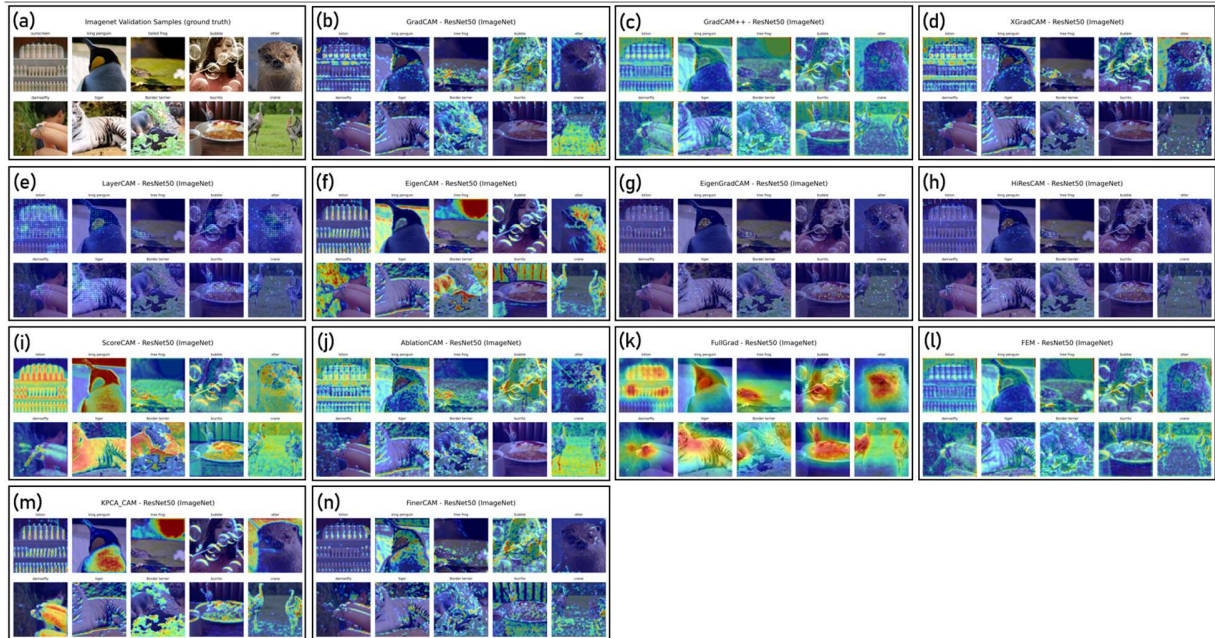


Figura 5.4. CAMs generados por distintos métodos a partir de las activaciones de salida de la etapa 1 de ResNet-50.
 (a) Imágenes originales, (b) GradCAM, (c) GradCAM++, (d) XGradCAM, (e) LayerCAM, (f) EigenCAM, (g) EigenGradCAM, (h) HiResCAM, (i) ScoreCAM, (j) AblationCAM, (k) FullGrad, (l) FEM, (m) KPCA-CAM, (n) FinerCAM.

Por el contrario, en la última etapa prácticamente todos los métodos logran enfocar correctamente la región discriminativa para la clasificación, destacando claramente el objeto de interés, como se ejemplifica en la Figura 5.5.

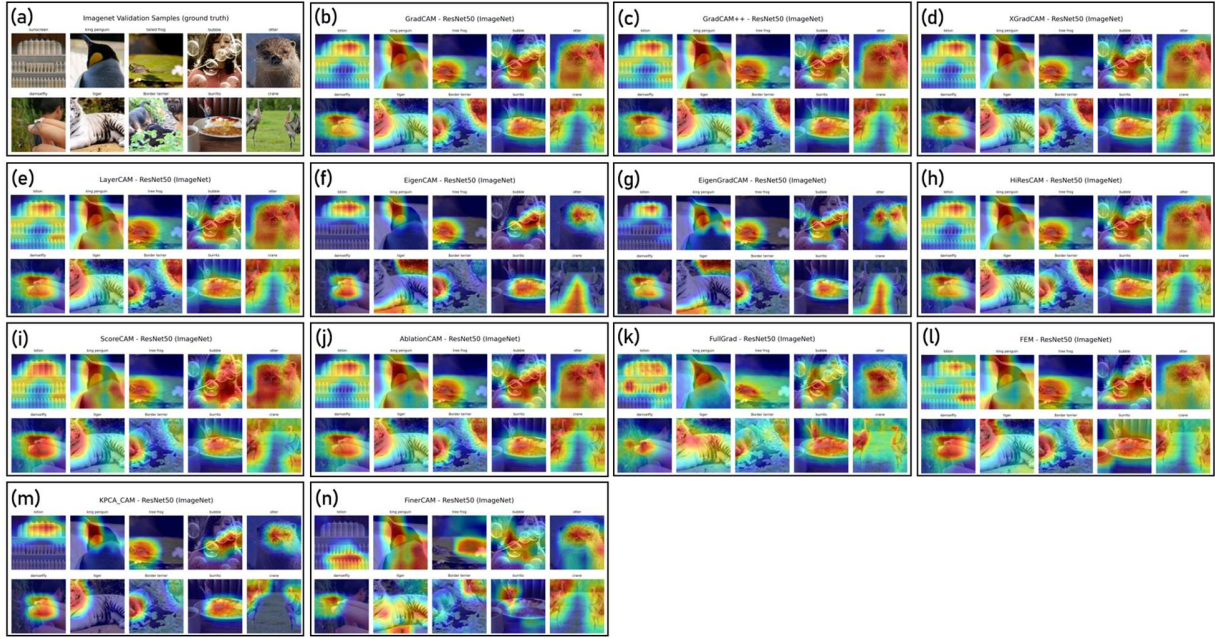


Figura 5.5. CAMs generados por distintos métodos a partir de las activaciones de salida de la etapa 4 de ResNet-50.
 (a) Imágenes originales, (b) GradCAM, (c) GradCAM++, (d) XGradCAM, (e) LayerCAM, (f) EigenCAM, (g) EigenGradCAM, (h) HiResCAM, (i) ScoreCAM, (j) Abl

El criterio utilizado para seleccionar los mapas representativos se basó en la claridad visual, precisión en la localización y nitidez con la que el objeto principal destacaba respecto al fondo y otras regiones no relevantes de la imagen. Además, considerando la jerarquía de características aprendidas por las CNNs, donde las capas iniciales suelen capturar información general como bordes y texturas, y las capas profundas representan semánticamente el objeto en su conjunto [73], [74], [75], se esperaba que los CAMs reflejaran este comportamiento: activaciones en bordes, texturas y colores en fases tempranas y centradas en el objeto de interés en etapas avanzadas.

Tomando esto en cuenta, EigenCAM se selecciona como un método prometedor debido a su capacidad consistente para reflejar el comportamiento esperado en las distintas etapas de ResNet-50. En las etapas iniciales, EigenCAM muestra activaciones amplias y dispersas, identificando características generales y patrones de bajo nivel, como bordes y texturas, coherente con la teoría del aprendizaje jerárquico en redes convolucionales. En contraste, en la etapa final, este método destaca las regiones discriminativas del objeto, proporcionando mapas con coherencia semántica. Los ejemplos que reflejan este comportamiento se pueden observar en la Figura 5.6. Esta combinación de comportamiento justifica su elección como un buen método para asignar importancia de filtros con el propósito de poda estructural.

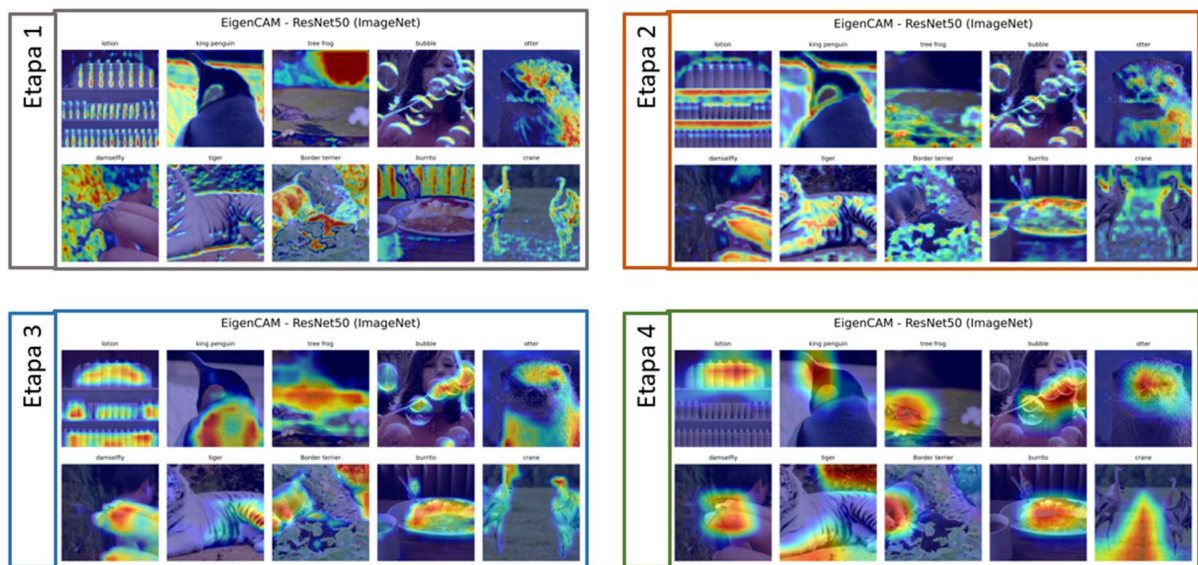


Figura 5.6. CAMs de las salidas de 4 etapas de ResNet-50 generados con EigenCAM.

5.2.2 Evaluación cuantitativa

Para complementar el análisis visual se aplicaron cuatro métricas cuantitativas de interpretabilidad, las cuales permiten evaluar objetivamente diferentes propiedades relevantes al contexto específico de poda estructural. Estas métricas son selectividad (*selectivity*), continuidad (*continuity*), enfoque (*focus*) y Nivel de fuerza de una explicación (*Level of Strength of Explanation*, LSE). A continuación, se describen detalladamente cada una de ellas y cómo se relacionan específicamente con el objetivo de seleccionar filtros relevantes en una red neuronal profunda.

5.2.2.1 Selectividad

La métrica de selectividad evalúa qué tan precisamente el método identifica las características más importantes para la predicción del modelo [76]. Su fundamento radica en la idea de que una perturbación en una característica altamente relevante producirá un cambio significativo en la predicción del modelo. Para medir la selectividad, se realiza lo siguiente:

1. Se ordenan las características desde las más relevantes hasta las menos relevantes según la explicación del método CAM.
2. Se enmascaran progresivamente las características más relevantes y, en cada paso, se mide el cambio en la predicción del modelo.

3. Se genera una curva en donde el eje de las abscisas representa la cantidad acumulada de características eliminadas y el eje las ordenadas representa el valor de la predicción para la clase objetivo.
4. Finalmente, se calcula el área bajo la curva (AUC). Un valor pequeño de AUC indica alta selectividad, ya que implica que la eliminación de características relevantes afecta significativamente la predicción.

Para ilustrar el cálculo de esta métrica, se generó un ejemplo visual mediante el proceso descrito anteriormente. En la Figura 5.7, se muestra cómo se realiza el procedimiento de enmascaramiento progresivo de los parches de 8x8 píxeles más relevantes de acuerdo con el mapa CAM generado. El tamaño de 8x8 píxeles ofrece un balance entre granularidad y eficiencia computacional. Se observa que al enmascarar gradualmente estas regiones, la probabilidad de la clase objetivo disminuye, representado en la curva.

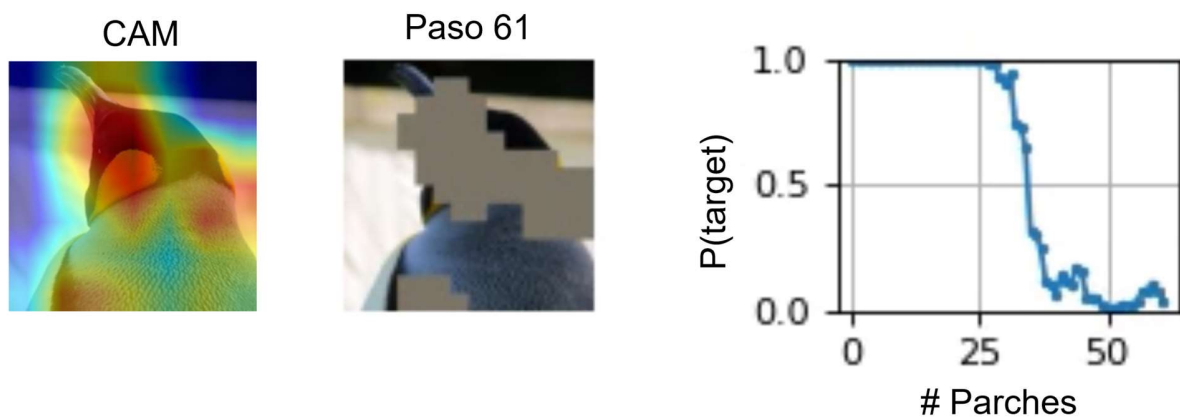


Figura 5.7. Ejemplo visual de procedimiento para obtención de métrica de selectividad.

En el contexto específico de poda estructural, una alta selectividad implica que el método CAM correspondiente identifica correctamente aquellos filtros cuyas activaciones tienen un mayor impacto en las decisiones del modelo. Por tanto, seleccionar filtros utilizando métodos CAM con valores bajos de la métrica de selectividad (AUC) permite eliminar efectivamente filtros poco relevantes, minimizando la pérdida en desempeño al comprimir la red neuronal.

5.2.2.2 Continuidad

La métrica de continuidad evalúa si pequeñas modificaciones en la entrada provocan igualmente pequeñas modificaciones en el mapa de relevancia generado por un método de explicación. Este comportamiento es deseable, ya que una propiedad esperada de un buen método explicativo es que sea una función continua, es decir, si dos entradas son similares, entonces sus explicaciones también deberían ser similares [77].

Para calcular esta métrica, se sigue el siguiente procedimiento:

1. Se toma una imagen original img y se genera una versión ligeramente perturbada $\overline{img}$, por ejemplo, mediante desplazamiento, ruido o rotación leve.
2. Se generan los mapas de relevancia $C_q^c(img)$ y $C_q^c(\overline{img})$ para ambas versiones utilizando un método CAM.
3. Se calcula la diferencia entre ambos mapas mediante la norma ℓ_1 , y se normaliza por la distancia entre las imágenes en el espacio de entrada con la norma ℓ_2 .

La métrica se define formalmente como:

$$Continuity^{(q)} = \max_{img \neq \overline{img}} \frac{\|C_q^c(img) - C_q^c(\overline{img})\|_1}{\|img - \overline{img}\|_2} \quad (5.4)$$

donde un valor pequeño de $Continuity^{(q)}$ indica alta continuidad, ya que el cambio en la explicación es proporcional o inferior al cambio en la entrada.

Para ilustrar esta métrica, en la Figura 5.8 se muestra un ejemplo en el que una imagen es desplazada progresivamente hacia la derecha, en intervalos de 50 píxeles. Cada imagen corresponde al mapa de activación de clase $C_q^c(img)$ generado por el método HiResCAM para la clase *king penguin*, considerando distintos desplazamientos. Se puede observar que, conforme la imagen se altera, los mapas de relevancia también se modifican. En un método con buena continuidad, estos cambios deberían ser suaves y coherentes con la alteración en la entrada. En cambio, si los mapas cambian de forma abrupta o inconsistente ante pequeños desplazamientos, esto indica una baja continuidad en la explicación.

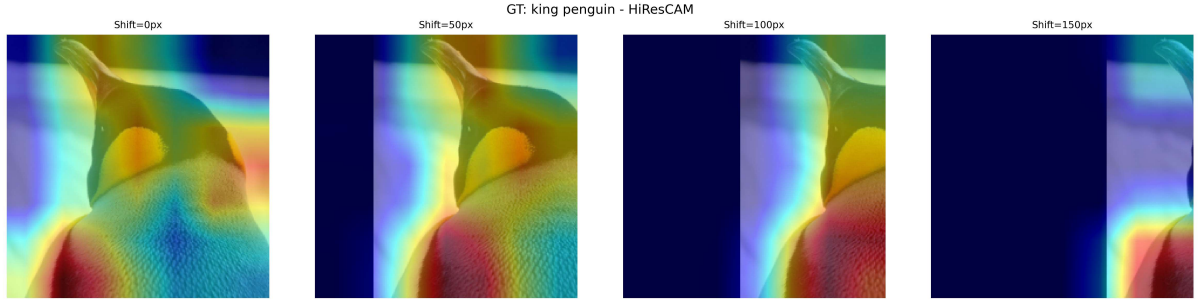


Figura 5.8. Ejemplo visual del cálculo de la métrica de continuidad.

En el contexto de la poda estructural, una buena continuidad sirve para asegurar que los filtros seleccionados como relevantes no dependan de activaciones inestables o sensibles a perturbaciones mínimas. Esto permite tomar decisiones más robustas sobre qué filtros conservar o eliminar durante el proceso de compresión del modelo.

5.2.2.3 Enfoque

La métrica de enfoque evalúa la capacidad de un método CAM para concentrar su relevancia específicamente en aquellas regiones que contienen objetos de la clase objetivo, evitando asignar relevancia a regiones distractoras o irrelevantes para dicha clase [78].

Para calcular esta métrica se sigue el siguiente procedimiento:

1. Se genera un mosaico $\mathbf{m}$ compuesto por cuatro subimágenes, dos de ellas pertenecientes a la clase objetivo c , y dos pertenecientes a otras clases diferentes, también llamadas imágenes distractoras.
2. Se obtiene el mapa de activación de clase $C_q^c(\mathbf{m})$ usando un método explicativo q .
3. Se calcula la proporción de relevancia positiva que se concentra únicamente en las subimágenes correspondientes a la clase objetivo dentro del mosaico, respecto a la relevancia total en el mosaico.

Formalmente, la métrica de enfoque se define como:

$$Focus_q^{(c)}(\mathbf{m}) = \frac{R_c(img_1) + R_c(img_2)}{R_c(\mathbf{m})} \quad (5.5)$$

Donde $R_c(img_i)$ es la suma de la relevancia positiva asignada por el método explicativo q hacia la clase objetivo c dentro de la subimagen img_i y $R_c(\mathbf{m})$ es la relevancia positiva total asignada hacia la clase objetivo en todo el mosaico $\mathbf{m}$.

Un valor de enfoque cercano a 1 indica que el método CAM asigna correctamente la mayoría de su relevancia únicamente a las subimágenes que pertenecen a la clase objetivo, mostrando así una alta capacidad de discriminación visual. Un valor cercano a 0.5 sugiere un comportamiento prácticamente aleatorio del método CAM, al no diferenciar claramente entre regiones relevantes e irrelevantes.

Para ilustrar esta métrica, en la Figura 5.9 se muestra un ejemplo visual del proceso descrito. El mosaico está compuesto por dos subimágenes que pertenecen a la clase *Schipperke*, que es la clase objetivo, y dos subimágenes distractoras, *Border collie* y *Helmet*. Se observa claramente cómo el método ScoreCAM concentra adecuadamente la mayor parte de la relevancia en las subimágenes correspondientes a la clase objetivo (parte superior derecha e inferior izquierda del mosaico), generando así un valor de enfoque de 0.735.



Figura 5.9. Ejemplo visual del cálculo de la métrica de enfoque.

Utilizar métodos CAM con un enfoque alto, en el contexto de la poda estructural, asegura que los filtros seleccionados para poda estén verdaderamente relacionados con características visuales discriminativas y no con patrones irrelevantes, lo que contribuye directamente a mantener un buen desempeño del modelo después del proceso de compresión.

5.2.2.4 Nivel de Fuerza de una Explicación (LSE)

La métrica de nivel de fuerza de una explicación (LSE) cuantifica hasta qué punto la explicación proporcionada por un método explicativo respalda específicamente a la clase predicha por el modelo en comparación con el resto de las clases posibles [79].

La intuición detrás de esta métrica es que si un método explicativo asigna altas relevancias a las regiones cuando estas se relacionan con la clase objetivo, y asigna bajas relevancias a las mismas regiones cuando se evalúan otras clases, entonces la explicación se considera fuerte y específica para la clase objetivo.

Para calcular la métrica LSE, se sigue el siguiente procedimiento:

1. Se obtiene el mapa de activación de clase $C_q^c(\mathbf{x})$ para la clase objetivo c usando un método explicativo q .
2. Se obtienen los mapas de activación $C_q^j(\mathbf{x})$ para cada una de las demás clases $j \neq c$.
3. Se seleccionan las regiones o características espaciales más relevantes usando un umbral ξ .
4. Finalmente, se calcula la relación entre la relevancia promedio asignada a la clase objetivo y la relevancia promedio asignada al resto de las clases en esas regiones seleccionadas:

$$LSE(\xi, c) = \frac{\frac{1}{|S_c(\xi)|} \sum_{i \in S_c(\xi)} r_{ic}}{\frac{1}{L-1} \sum_{j=1, j \neq c}^L \left(\frac{1}{|S_c(\xi)|} \sum_{i \in S_c(\xi)} r_{ij} \right)}, \quad (5.6)$$

donde:

- L es el número total de clases.
- r_{ik} es el valor de relevancia para la clase k en la característica o región i , obtenida mediante el método explicativo q .

- ξ es un umbral que define el criterio para seleccionar las características más relevantes dentro del mapa.
- $S_c(\xi)$ es el conjunto de índices espaciales o características cuya relevancia hacia la clase objetivo c supera el umbral ξ .

Un valor alto de LSE indica que la explicación generada es específica y está claramente orientada hacia la clase objetivo, mientras que valores cercanos o inferiores a 1 indican explicaciones poco específicas, que no destacan claramente a la clase objetivo frente a otras clases. En la Figura 5.10 se muestra un ejemplo en el que se comparan los mapas generados por un método explicativo para la clase objetivo *Macaw* con los mapas generados para otras clases diferentes. Si la relevancia es considerablemente mayor en el mapa correspondiente a la clase objetivo y notablemente menor en los mapas de las otras clases, resultará en un valor alto de LSE.

$$LSE(\xi, c) = \frac{\frac{1}{|S_c(\xi)|} \sum_{i \in S_c(\xi)} r_{ic}}{\frac{1}{L-1} \sum_{j=1, j \neq c}^L \left(\frac{1}{|S_c(\xi)|} \sum_{i \in S_c(\xi)} r_{ij} \right)} = \frac{\text{Macaw (0.85)}}{\text{German shepherd (0.0)} + \text{Australian terrier (0.01)} + \text{Tiger cat (0.48)} + \dots + \text{Bee eater (0.88)}} \approx \frac{0.85}{(0.0+0.01+0.48+\dots+0.88)/999} = \frac{0.85}{0.22} \approx 3.82$$

Figura 5.10. Ejemplo visual del cálculo de la métrica LSE.

En cuanto al contexto de la poda estructural, un valor alto de LSE se traduce en que los filtros seleccionados por el método explicativo están realmente ligados a la discriminación efectiva de la clase objetivo y no responden de forma ambigua a otras clases. Esto permite identificar los filtros relevantes que deben preservarse durante el proceso de poda.

5.2.3 Metodología para la evaluación comparativa de métodos XAI

Con el fin de seleccionar objetivamente el método de XAI más adecuado para guiar el proceso de poda estructural en redes neuronales profundas, se diseñó una metodología comparativa que involucró la evaluación de múltiples métodos CAM mediante las métricas cuantitativas presentadas en la sección anterior.

El protocolo general de evaluación consiste en la evaluación de un total de 13 métodos CAM diferentes, empleando 10 imágenes distintas seleccionadas aleatoriamente del conjunto de validación de ImageNet-1k, estas 10 imágenes se muestran en la Figura 5.11. Con esto, se buscó reducir posibles sesgos asociados a características específicas de imágenes individuales, buscando así resultados generalizables y robustos.



Figura 5.11. Imágenes seleccionadas aleatoriamente del conjunto de datos ImageNet-1k con su respectiva clase.

En particular, se obtuvieron métricas de interpretabilidad (selectividad, continuidad, enfoque y LSE) en cuatro etapas distintas de la arquitectura ResNet-50, tal como se hizo en la evaluación cualitativa, las etapas de las que se generaron los CAMs se muestran en la Figura 5.3. El uso de múltiples etapas permite analizar y comparar cómo se comportan los métodos CAM en diferentes niveles de abstracción dentro de la red neuronal.

En el caso específico de la métrica de enfoque, únicamente se reportaron los resultados obtenidos en la última etapa (etapa 4). Esta decisión se tomó debido a que, en las etapas iniciales (1, 2 y 3), las métricas obtenidas fueron cercanas a 0.5, lo que implica que los métodos CAM no fueron capaces de distinguir claramente entre las subimágenes objetivo y las distractoras, como se ilustra en la Figura 5.12. Esto es esperado, dado que capas iniciales suelen capturar características más generales y de bajo nivel, mientras que capas profundas extraen características semánticamente más específicas, relevantes para la identificación visual precisa.

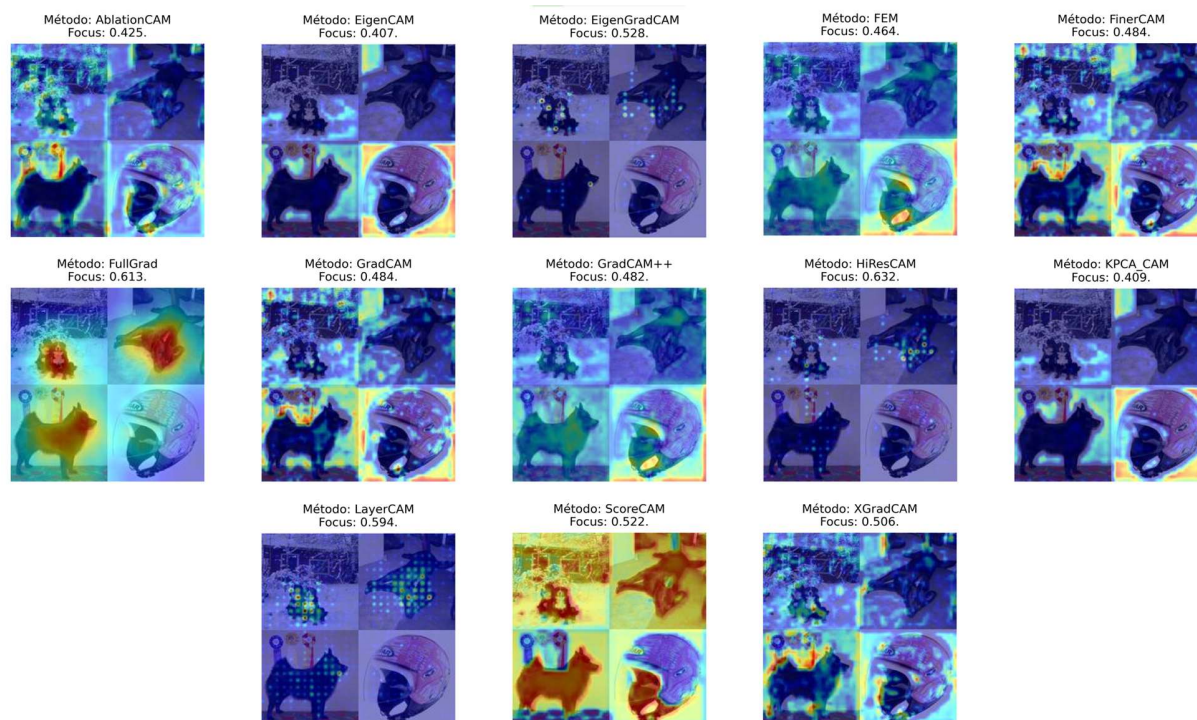


Figura 5.12. Ejemplos de CAMs generados a partir de mosaicos para calcular métrica de enfoque.

Cada método CAM fue evaluado utilizando las 10 imágenes mencionadas, generando mapas de activación para cada imagen y cada etapa evaluada. Posteriormente, se calcularon las cuatro métricas (selectividad, continuidad, enfoque y LSE) para cada combinación método-imagen-etapa. Para el cálculo de selectividad se decidió utilizar parches de 8x8 píxeles, en el caso del cálculo de continuidad se modificó la imagen original haciendo un desplazamiento de 50 píxeles a la derecha y se compararon 3 imágenes modificadas con la original y para el cálculo del LSE se tomó un umbral $\xi = 0.5$.

A partir de los resultados individuales, se obtuvieron tablas resumen para cada métrica y cada etapa, calculando promedios y desviaciones estándar por método CAM, lo que permitió determinar con claridad cuál método presentó un mejor desempeño global. Un ejemplo representativo de dichas tablas, específicamente para la métrica de continuidad en la etapa 4, se presenta en la Tabla 5.1. El resto de tablas resumen detalladas se pueden consultar en anexos. El sombreado de color representa la magnitud relativa de los valores, los tonos más claros corresponden a mejores resultados, mientras que los tonos más oscuros indican resultados menos favorables.

Tabla 5.1. Resultados de la métrica de continuidad para cada método CAM evaluado en 10 imágenes (Etapa 4 de ResNet-50). Valores menores son mejores.

	Imagen 1	Imagen 2	Imagen 3	Imagen 4	Imagen 5	Imagen 6	Imagen 7	Imagen 8	Imagen 9	Imagen 10	Media↓	Std↓	Rank↓
AblationCAM	23.4756	23.7165	26.4399	21.5861	39.8684	31.7482	16.7366	24.3709	22.1386	42.1484	27.2229	7.7835	6
EigenCAM	16.0307	12.4547	28.9564	12.3326	23.5199	21.6485	23.4167	18.4876	11.9801	27.8644	19.6692	6.0484	1
EigenGradCAM	17.0351	14.3125	27.9451	15.6387	25.7856	25.4288	22.3878	20.2979	14.1053	25.8944	20.8831	5.037	2
FEM	24.4736	22.8963	30.7185	14.447	34.7121	29.3296	18.9794	23.5198	31.8748	42.4166	27.3368	7.7219	7
FinerCAM	24.3466	23.7232	27.528	23.6482	40.0286	30.369	16.6409	24.3014	20.6276	47.6548	27.8868	8.8563	10
FullGrad	23.8511	20.9893	23.6689	26.9018	30.7256	28.0813	17.0777	16.321	24.5518	29.3927	24.1561	4.6426	4
GradCAM	24.3466	23.7232	27.528	23.6482	40.0286	30.369	16.6409	24.3014	20.6276	47.6548	27.8868	8.8563	10
GradCAM++	20.4276	23.4788	28.0354	21.5041	39.5006	31.5921	16.8199	23.1726	23.1011	46.3854	27.4018	8.7857	9
HiResCAM	24.3466	23.7232	27.528	23.6482	40.0286	30.369	16.6409	24.3014	20.6276	47.6548	27.8868	8.8563	10
KPCA_CAM	17.4675	18.4777	31.6121	11.9207	24.8229	24.3037	17.1579	20.632	13.9378	45.1319	22.5464	9.2878	3
LayerCAM	19.8746	23.4895	28.1561	20.9211	39.5852	31.9421	16.7894	23.3357	24.0866	45.775	27.3955	8.7253	8
ScoreCAM	23.0046	20.4058	25.3976	21.4886	36.3855	32.2369	17.0811	24.4704	23.0153	37.7388	26.1225	6.6044	5
XGradCAM	24.3466	23.7232	27.528	23.6482	40.0286	30.3691	16.6409	24.3014	20.6276	47.6548	27.8868	8.8563	13

Una vez obtenidos los valores promedio por cada métrica, imagen y etapa, se procedió a realizar rankings individuales para cada métrica, ordenando los métodos según sus valores numéricos. En función de la métrica específica, un valor numérico alto o bajo implica un mejor desempeño, por ejemplo, un valor bajo en selectividad indica un mejor método CAM. Posteriormente, se sumaron las posiciones en los rankings obtenidas por cada método para cada métrica y cada etapa, obteniendo así una evaluación general del desempeño relativo de cada método CAM.

Los resultados de estos rankings individuales se resumen en la Tabla 5.2, permitiendo visualizar fácilmente qué métodos destacaron en cada métrica individual.

Tabla 5.2. Rankings de los métodos CAM según cada métrica considerando todas las etapas.

Selectivity			Continuity			Focus			LSE		
Método	Suma ranks	Rank final	Método	Suma ranks	Rank final	Método	Suma ranks	Rank final	Método	Suma ranks	Rank final
HiResCAM	8	1	EigenGradCAM	6	1	EigenGradCAM	1	1	EigenGradCAM	7	1
LayerCAM	11	2	HiResCAM	15	2	GradCAM	2	2	HiResCAM	8	2
FullGrad	19	3	LayerCAM	18	3	XGradCAM	2	2	FinerCAM	11	3
XGradCAM	22	4	XGradCAM	21	4	HiResCAM	2	2	GradCAM	11	3
GradCAM++	24	5	GradCAM	25	5	FinerCAM	2	2	XGradCAM	12	4
FEM	25	6	FinerCAM	25	5	AblationCAM	3	3	AblationCAM	19	5
ScoreCAM	26	7	GradCAM++	28	6	ScoreCAM	4	4	ScoreCAM	25	6
EigenGradCAM	31	8	FEM	30	7	GradCAM++	5	5	FullGrad	29	7
GradCAM	31	8	KPCA_CAM	30	7	LayerCAM	6	6	LayerCAM	31	8
AblationCAM	31	8	ScoreCAM	31	8	FullGrad	7	7	GradCAM++	37	9
EigenCAM	39	9	AblationCAM	33	9	EigenCAM	8	8	EigenCAM	38	10
FinerCAM	43	10	FullGrad	35	10	FEM	9	9	FEM	38	10
KPCA_CAM	47	11	EigenCAM	37	11	KPCA_CAM	10	10	KPCA_CAM	38	10

Dado que las métricas utilizadas presentan distinta importancia en el contexto específico de poda estructural, se decidió asignar ponderaciones a cada métrica según su relevancia relativa. Esta ponderación también consideró ciertas limitaciones observadas durante la evaluación. Por

ejemplo, algunos métodos con aparentemente buenos resultados en continuidad generaron mapas CAM con muy baja activación, lo cual reducía artificialmente las diferencias numéricas en los mapas ante pequeñas perturbaciones. Este fenómeno se ilustra claramente en la Figura 5.13, donde se comparan métodos con buena continuidad (poca activación) frente a métodos con pobre continuidad (alta activación).

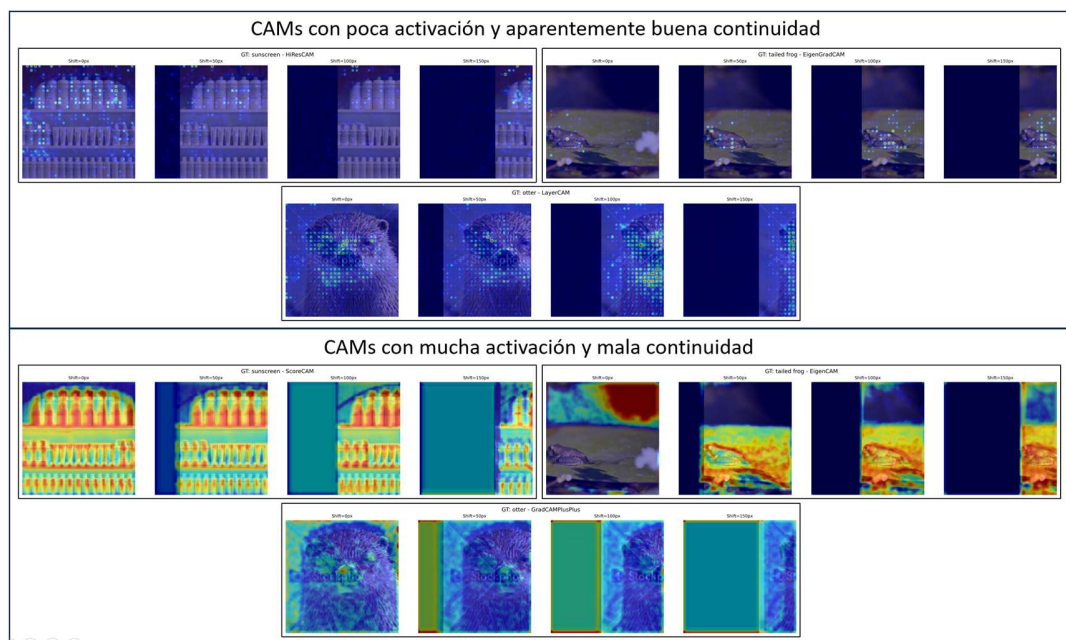


Figura 5.13. Comparación de CAMs con bajas y altas activaciones.

La Tabla 5.3 resume las ponderaciones asignadas a cada métrica y la justificación detrás de estas decisiones. En cuanto a la asignación de ponderaciones, la métrica Focus recibió un valor bajo (0.1) debido a que únicamente resulta útil para analizar la etapa 4 del modelo, sin ser aplicable de manera general a todas las etapas. La métrica Continuity obtuvo una ponderación de 0.2, ya que, si bien en teoría una buena continuidad implica consistencia en la ponderación de los filtros, es decir, que se resalten las mismas características incluso cuando el objeto se desplaza, durante los experimentos se observaron ciertas inconsistencias en su cálculo. En contraste, las métricas Selectivity y LSE no presentaron deficiencias en su evaluación y aportan información robusta para la poda estructural, por lo que se les asignó una ponderación más alta y equivalente (0.35 cada una).

Tabla 5.3. Métricas de evaluación de métodos de XAI y su ponderación.

Métrica	¿Por qué importa al podar?	Ponderación
Selectivity	Un filtro que recibe mucho peso debería ser crítico para la salida. Si el CAM tiene una buena ponderación se cumple esto.	0.35
Continuity	Existe consistencia en la ponderación de los filtros.	0.2
Focus	Si el CAM dispersa la relevancia, asignará peso a filtros que no aportan a la clase objetivo.	0.1
LSE	Al podar importa conservar filtros que diferencian la clase correcta de las otras.	0.35

Finalmente, con las ponderaciones definidas, se calculó un puntaje global ponderado para cada método CAM, obteniendo así un ranking final. Este ranking final, presentado en la Tabla 5.4, reveló claramente que el método HiResCAM obtuvo el mejor desempeño, justificando su selección como el método más apropiado para utilizarse en la poda estructural informada por métodos XAI.

Tabla 5.4. Ordenamiento de métodos de XAI con ponderación de rankings.

Método	Puntuación total	Rank global
HiResCAM	1.65	1
EigenGradCAM	3.45	2
XGradCAM	3.8	3
LayerCAM	4.7	4
GradCAM	5.05	5
FinerCAM	6.1	6
FullGrad	6.2	7
ScoreCAM	6.55	8
GradCAM++	6.6	9
AblationCAM	6.65	10
FEM	7.9	11
EigenCAM	10	12
KPCA_CAM	10.1	13

5.3 Selección y descripción de métodos XAI

Una vez analizados cuantitativa y cualitativamente los resultados de los 13 métodos CAM evaluados, se observaron ciertas características comunes entre aquellos métodos que obtuvieron los mejores resultados en el ranking ponderado global: HiResCAM, EigenGradCAM, XGradCAM y LayerCAM. Estos métodos integran de manera efectiva la información del gradiente, lo que les permite captar características relevantes con una mayor precisión y consistencia a lo largo de

múltiples etapas de la red neuronal. Específicamente, todos estos métodos pertenecen a la familia CAM basada en gradientes, donde la relevancia de cada región se determina mediante el análisis de la sensibilidad del resultado final (la clase predicha) con respecto a las activaciones internas del modelo.

Una característica notable en estos métodos es que generan mapas de relevancia más focalizados y menos dispersos en comparación con otros métodos evaluados. Por ejemplo, EigenGradCAM y HiResCAM destacan por generar mapas con activaciones más concentradas espacialmente debido a la forma específica en que integran el gradiente con las activaciones (Figura 5.4 (g) y (h)). En particular, EigenGradCAM realiza un análisis de componentes principales (PCA) sobre las activaciones ponderadas por gradientes, lo cual permite extraer el patrón de activación global más significativo asociado a la clase predicha, resultando en mapas visualmente claros y menos ruidosos. Por otro lado, HiResCAM calcula la contribución individual de cada posición espacial multiplicando directamente el gradiente local por la activación correspondiente, preservando así información de alta resolución y evitando que regiones no relevantes sean resaltadas incorrectamente, aunque esto pueda dar lugar a mapas con menos activaciones dispersas en las etapas iniciales.

Por su parte, métodos como XGradCAM y LayerCAM tienden a producir activaciones más detalladas y coherentes en etapas tempranas e intermedias debido a la asignación de pesos específicos para cada posición espacial dentro de los mapas de características (Figura 5.4 (d) y (e)). XGradCAM ajusta sus pesos mediante una ponderación explícita de los gradientes basada en la magnitud local de las activaciones, cumpliendo así criterios teóricos como la sensibilidad y conservación. Esto lo hace particularmente efectivo en reflejar con precisión qué regiones del mapa de características contribuyen significativamente al resultado final del modelo. LayerCAM, por otro lado, enfatiza aún más esta precisión espacial mediante la asignación explícita de pesos locales definidos por los gradientes positivos, lo cual es especialmente útil en capas intermedias de la red, resultando en mapas altamente detallados capaces de delinear claramente incluso pequeños detalles relevantes del objeto objetivo.

Sin embargo, durante el análisis comparativo también se identificaron discrepancias entre los resultados cuantitativos y cualitativos. Particularmente, destaca el caso del método EigenCAM,

variante anterior a EigenGradCAM, que desde un punto de vista cualitativo generó mapas visualmente atractivos, claros y bien definidos desde la perspectiva humana (Figura 5.6). No obstante, EigenCAM obtuvo resultados cuantitativos desfavorables en comparación con otros métodos. Por el contrario, HiResCAM, el método que obtuvo mejores resultados en términos cuantitativos, presenta visualmente mapas con activaciones escasas y poco informativas en etapas iniciales (etapas 1, 2 y 3), como se observa en la Figura 5.14.

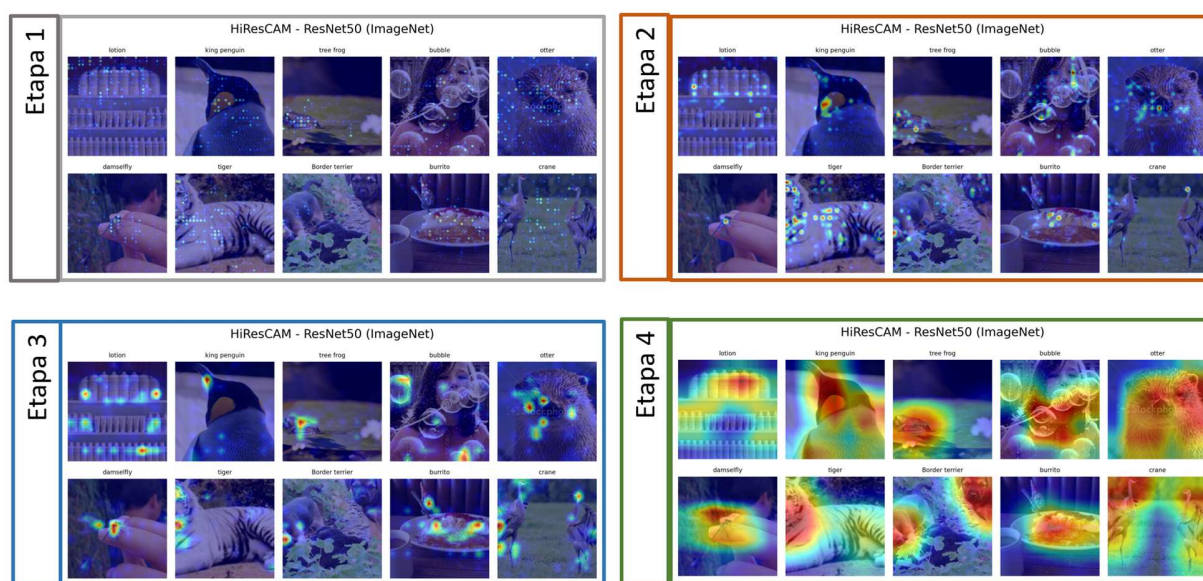


Figura 5.14. CAMs de las salidas de 4 etapas de ResNet-50 generados con HiResCAM.

Este comportamiento contradictorio puede entenderse mejor analizando el procedimiento utilizado para calcular la métrica de selectividad, ilustrado en la Figura 5.15. Al comparar el desempeño de EigenCAM y HiResCAM en dicha métrica, se observa claramente que, aunque HiResCAM genera mapas con pocas activaciones, dichas activaciones corresponden precisamente a las regiones más críticas para la predicción del modelo debido a su rigor en la integración local del gradiente con las activaciones. Por lo tanto, al remover secuencialmente dichas regiones mediante parches, a pesar de ser pocos, la probabilidad de la clase objetivo desciende rápidamente. En cambio, EigenCAM genera activaciones visualmente llamativas para humanos como bordes y texturas, pero la eliminación de esas áreas no afecta significativamente la predicción del modelo, indicando que esas activaciones no son críticas para la decisión real del modelo.

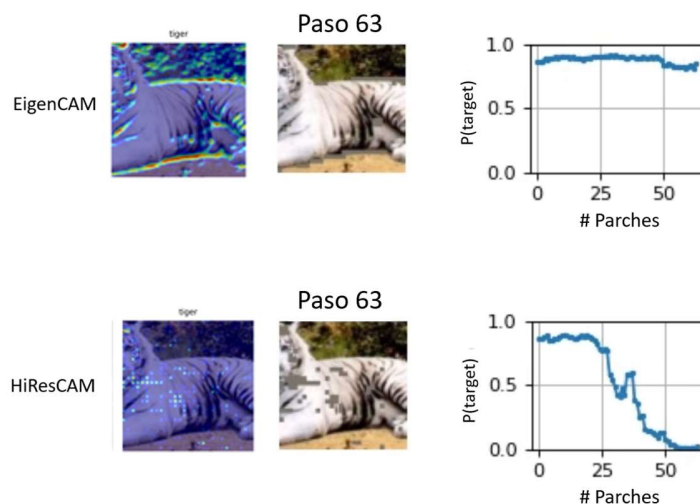


Figura 5.15. Comparación de Selectividad de EigenCAM y HiResCAM.

Este fenómeno sugiere que la intuición visual humana respecto a qué regiones de una imagen deberían ser relevantes para una predicción no necesariamente coincide con las características que el modelo considera críticas para dicha decisión. Otra posible razón radica en la existencia de dependencias jerárquicas entre filtros y características en diferentes etapas de la red neuronal, es decir, una activación pequeña en etapas iniciales puede tener gran influencia al activar posteriormente filtros en etapas más profundas, generando un impacto significativo en la predicción final, aunque visualmente dicha activación parezca irrelevante a ojos humanos. Métodos como HiResCAM y LayerCAM, al operar explícitamente a nivel local, logran capturar estos detalles con mayor eficacia, lo que explica sus buenos desempeños cuantitativos en métricas como Selectividad o Continuidad.

Considerando esta observación, se tomó la decisión de seleccionar tanto HiResCAM (basado en datos cuantitativos) como EigenCAM (basado en percepción cualitativa humana) para ser comparados posteriormente en términos del desempeño que generan al ser utilizados en la poda estructural. El objetivo de esta comparación es determinar empíricamente cuál criterio resulta más efectivo para guiar el proceso de compresión de modelos, la intuición visual humana o las métricas cuantitativas.

Finalmente, a continuación se presenta una descripción detallada del funcionamiento específico de los métodos HiResCAM y EigenCAM, los cuales serán utilizados posteriormente en la metodología de poda propuesta en el CAPÍTULO VI.

5.3.1 EigenCAM

El método EigenCAM fue propuesto por Bany y Yeasin para generar CAMs mediante el uso de componentes principales, evitando así la necesidad de utilizar gradientes o pesos derivados del *backpropagation*, como ocurre en métodos como Grad-CAM o CAM original [67]. Esta característica hace que EigenCAM sea especialmente robusto frente a errores de predicción y al ruido en los gradientes, al depender únicamente de la información extraída por la red convolucional, sin requerir propagación hacia atrás del gradiente.

EigenCAM parte de la premisa de que una CNN mapea la imagen de entrada a un espacio de representación de características en el que se pueden discriminar clases. En dicho espacio, las características más relevantes para la clasificación estarán alineadas en direcciones específicas que capturan la mayor varianza de los datos, un ejemplo práctico se puede observar en la Figura 5.16. De esta manera, las direcciones principales obtenidas mediante PCA maximizan la varianza global, al tomar el primer componente principal, la discriminación de características se facilita.

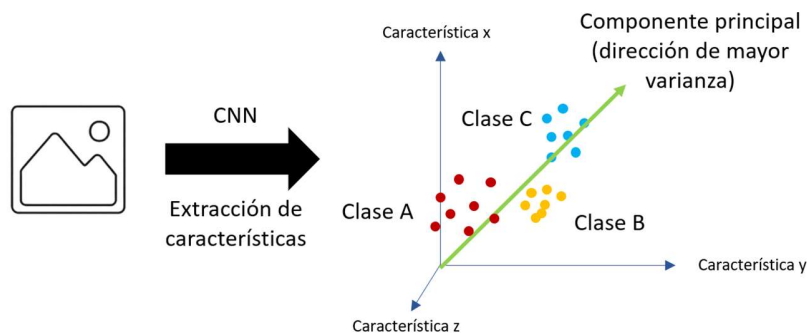


Figura 5.16. Mapeo de características a un espacio discriminativo.

El procedimiento específico para obtener EigenCAM se resume en los siguientes pasos:

1. Obtención de activaciones convolucionales: Se extraen las activaciones de una capa convolucional específica, representadas por un tensor $M \in \mathbb{R}^{C \times H \times W}$, donde C es el número de canales o mapas de activación y H y W son las dimensiones espaciales.
2. Redimensionamiento en matriz de píxeles: Las activaciones se reordenan en una matriz bidimensional de tamaño $C \times N$, donde $N = H \times W$. De esta manera, cada canal de activación se convierte en un vector.

$$O = \text{reshape}(M, (C, HW)) \in \mathbb{R}^{C \times N}, N = H \times W \quad (5.7)$$

3. Descomposición en valores singulares (SVD): Se aplica SVD a la matriz O , obteniendo tres matrices U , Σ , V :

$$O = U \Sigma V^* \quad (5.8)$$

En esta factorización las columnas de la matriz U son los vectores singulares izquierdos, los cuales representan la relevancia de cada canal de activación. Σ es una matriz diagonal con valores singulares que cuantifican la varianza explicada por cada componente y las columnas de la matriz V son vectores singulares derechos, que indican cómo se proyectan los píxeles en el espacio de representación aprendido.

4. Uso del primer vector propio: El primer vector singular izquierdo $U_{[:,0]}$ asociado al mayor valor singular es el más significativo, pues representa la dirección principal que captura la máxima varianza en el espacio de características. Este vector, al contener pesos que indican la relevancia relativa de cada canal convolucional, se utiliza como coeficientes para ponderar linealmente las activaciones:

$$\mathbf{w} = U_{[:,0]} \quad (5.9)$$

5. Obtención final del EigenCAM: Finalmente, se genera el mapa de activación de clase utilizando esta ponderación lineal:

$$C_{\text{EigenCAM}}^c = \text{reshape}(\mathbf{w}^* O, (H, W)) \quad (5.10)$$

Debido a que EigenCAM utiliza únicamente la información del primer vector propio para ponderar la importancia relativa de los canales, tiene la ventaja de identificar consistentemente regiones críticas para la predicción de clase, independientemente de la exactitud del clasificador final. Esto también explica por qué EigenCAM presenta robustez ante cambios introducidos por ruido en los gradientes o errores en capas totalmente conectadas, ya que la ponderación de las activaciones no depende directamente del gradiente o del resultado final de la predicción, sino

exclusivamente de la estructura de la información espacial y semántica aprendida por las capas convolucionales.

5.3.2 High-Resolution Class Activation Mapping (HiResCAM)

HiResCAM (*High-Resolution Class Activation Mapping*) fue desarrollado por Draelos y Carin para superar una limitación específica del popular método Grad-CAM. Esta limitación radica en el proceso de promediado global de los gradientes realizado por Grad-CAM, que ocasiona la pérdida de información local y, en consecuencia, mapas de relevancia menos precisos y potencialmente engañosos [62].

La intuición principal detrás de HiResCAM es evitar la simplificación del gradiente mediante su promediado, tal como lo hace GradCAM, lo que permite conservar información detallada y precisa de la relevancia espacial de cada canal. Así, HiResCAM genera mapas con mayor resolución espacial y una correspondencia exacta con las regiones específicas que el modelo realmente utiliza para realizar sus predicciones.

Formalmente, para obtener el mapa de relevancia de la clase objetivo c , HiResCAM calcula directamente el producto elemento a elemento entre el mapa de características y el gradiente del puntaje de la clase objetivo con respecto a dicho mapa. Matemáticamente, esto se expresa como:

$$C_{HiResCAM}^c(x, y) = \sum_k \frac{\partial y^c}{\partial m_k^{(L)}(x, y)} \square m_k^{(L)}(x, y) \quad (5.11)$$

Donde $m_k^{(L)}(x, y)$ representa la activación del mapa de características del canal k en la posición espacial específica (x, y) de la capa L , y $\frac{\partial y^c}{\partial m_k^{(L)}(x, y)}$ representa el gradiente del puntaje de la clase objetivo respecto a dicha activación.

Como se observa en la Figura 5.17, la diferencia fundamental entre HiResCAM y Grad-CAM radica en que HiResCAM conserva la información espacialmente detallada del gradiente sin simplificarla mediante promedios. Esto permite captar relevancias locales específicas con mayor precisión. Por otro lado, Grad-CAM, al promediar los gradientes, suaviza las relevancias a nivel

espacial, perdiendo detalles importantes, además de ignorar aquellos gradientes que tienen valores negativos.

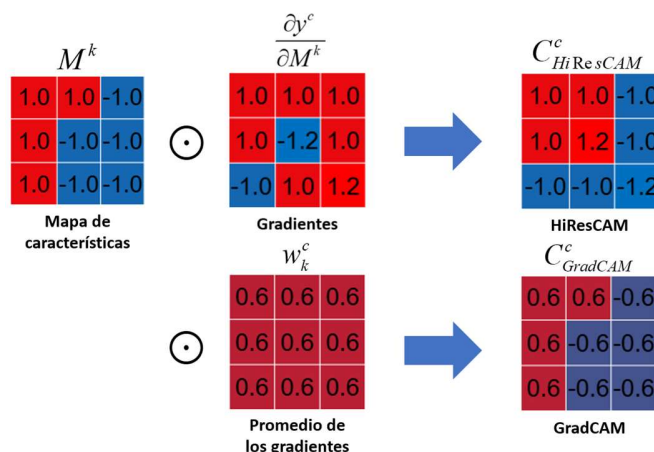


Figura 5.17. Representación de cálculo de HiResCAM y GradCAM [62].

La consideración de gradientes negativos es otra característica notable de HiResCAM, pues no ignora esta información. Mientras Grad-CAM utiliza la función de activación ReLU al final para eliminar relevancias negativas, HiResCAM aprovecha también estos valores negativos para proporcionar una visualización más fiel de lo que el modelo realmente considera relevante, ya sea de manera positiva o negativa, para la clase objetivo.

Esta capacidad de preservar información detallada y local puede resultar valiosa para tareas como la poda estructural, pues permite identificar con precisión los filtros y regiones específicas que más aportan o perjudican la predicción del modelo. Como se demostró previamente en los análisis cuantitativos, HiResCAM genera mapas menos vistosos desde una perspectiva visual humana, pero notablemente más efectivos desde la perspectiva del modelo, reflejando fielmente sus cálculos internos.

CAPÍTULO VI. METODOLOGÍA PROPUESTA DE PODA ESTRUCTURAL BASADA EN XAI

En este capítulo se propone una metodología que aprovecha técnicas de XAI para estimar la importancia de filtros convolucionales de manera directa, precisa e interpretable. Concretamente, se utilizarán los métodos EigenCAM y HiResCAM, identificados en capítulos anteriores como métodos efectivos para explicar las decisiones internas del modelo mediante CAMs. La hipótesis es que estos métodos, al captar información valiosa sobre las regiones y estructuras del modelo que contribuyen significativamente a la predicción, pueden proporcionar una base para establecer criterios eficaces de poda estructural.

Para implementar la metodología propuesta, se integran estos métodos XAI dentro del marco de trabajo ofrecido por el algoritmo de poda estructural iterativa presente en el *framework* TorchPruning. Este algoritmo permite realizar podas controladas, considerando dependencias estructurales internas de la red neuronal, para asegurar integridad funcional tras la eliminación de estructuras redundantes o de baja relevancia.

En las siguientes secciones, se describirá detalladamente el algoritmo general utilizado para la poda estructural, la forma específica en que EigenCAM y HiResCAM se adaptan para estimar la importancia de filtros convolucionales, así como consideraciones adicionales y estrategias prácticas para garantizar robustez y generalización del criterio propuesto.

6.1 Algoritmo general de poda estructural iterativa

La poda estructural iterativa es una estrategia efectiva para reducir significativamente la complejidad computacional y espacial de modelos de aprendizaje profundo mediante la eliminación sistemática de estructuras redundantes o poco relevantes del modelo original. La metodología propuesta en este trabajo emplea un procedimiento iterativo diseñado para alcanzar un factor de aceleración específico, manteniendo simultáneamente un alto desempeño predictivo.

Esta metodología se implementa utilizando el *framework* TorchPruning, que permite la poda estructurada segura de redes neuronales convolucionales mediante la identificación explícita de dependencias internas entre estructuras del modelo.

El procedimiento general propuesto se presenta de forma resumida en el Algoritmo 1, en el que se muestra el flujo completo de la metodología propuesta. A continuación, se describen de manera detallada las etapas involucradas en este procedimiento:

Algoritmo 1: Poda estructural iterativa global

Requiere: Modelo original $f(x;W)$, factor de aceleración deseado A (> 1), estimador de importancia I , conjunto de datos D , máximo número de iteraciones de poda N .

- | | |
|--|---|
| <p>1: $\hat{f}(x;W) = f(x;W)$, $n = 1$</p> <p>2: $\text{FLOPs}_{base} = \text{FLOPs}(f(x;W))$</p> <p>3: $DG \leftarrow \text{ConstruirGrafoDependencias}(f(x;W), x \in D)$</p> <p>3: Mientras $\frac{\text{FLOPs}(f(x;W))}{\text{FLOPs}(\hat{f}(x;W))} < A$ y $n \leq N$ hacer:</p> <p style="padding-left: 40px;">4: $S \leftarrow \text{EstimarImportancia}(f(\circ;W); DG; D; I)$</p> <p style="padding-left: 40px;">5: $n_{podar} = \text{NoCanales}_{actuales} - \text{NoCanales}_{iniciales} \left(1 - \frac{1}{N}\right)$</p> <p style="padding-left: 40px;">6: $idx_{poda} \leftarrow \text{argsort}(S)[n_{podar}]$</p> <p style="padding-left: 40px;">7: $\hat{f}(x;W) \leftarrow \text{Podar}(\hat{f}(x;W); idx_{podar})$</p> <p style="padding-left: 40px;">8: $n \leftarrow n + 1$</p> <p>Fin mientras</p> <p>9: Devolver: $\hat{f}(x;W)$</p> | <p>➤ Inicialización de modelo a podar y número de iteraciones.</p> <p>➤ Cálculo inicial de FLOPs.</p> <p>➤ Inicializa un grafo de dependencias DG con el modelo $f(x;W)$ y una entrada de ejemplo.</p> <p>➤ Calcula la importancia para cada estructura usando el estimador de importancia I.</p> <p>➤ Calcula el número de canales a podar del modelo.</p> <p>➤ Obtener índices de estructuras con menor importancia global.</p> <p>➤ Poda física: se reconstruyen capas dependientes, excluyendo las estructuras menos importantes.</p> <p>➤ Incrementa iteración.</p> <p>➤ Modelo compacto listo para re-entrenamiento.</p> |
|--|---|
-

Inicialmente, se asigna el modelo original $f(x;W)$ al modelo inicial que será podado representado como $\hat{f}(x;W)$. Además, se establece un contador de iteraciones para controlar las iteraciones del algoritmo.

Posteriormente, se calcula la cantidad inicial de FLOPs necesarias para realizar inferencias con el modelo original. Este cálculo inicial sirve como referencia para evaluar posteriormente si el modelo podado cumple con el factor de aceleración deseado.

Previo al comienzo del procedimiento iterativo, es necesario construir un grafo de dependencias (DG) [80]. Este grafo permite identificar y gestionar las dependencias estructurales internas del modelo original, garantizando que durante la poda no se eliminen estructuras cuya eliminación causaría inconsistencias arquitectónicas.

Con el grafo de dependencias construido, se inicia el proceso iterativo de poda. En cada iteración, se evalúa si se ha alcanzado el factor de aceleración deseado y si no se ha excedido el máximo número de iteraciones establecidas. Si estas condiciones se cumplen, se continúa el proceso de poda.

Dentro de cada iteración, el paso más importante es la estimación de la importancia de las estructuras internas del modelo. Esta etapa es crucial, ya que la calidad del criterio utilizado para evaluar la importancia tiene un impacto directo sobre el desempeño final del modelo podado. En este capítulo se hace especial énfasis en criterios de importancia basados en técnicas de XAI, específicamente EigenCAM y HiResCAM. Sin embargo, el procedimiento general presentado es suficientemente flexible para utilizar otros criterios de importancia revisados previamente en el capítulo 4, tales como criterios basados en magnitudes, activaciones o gradientes.

Una vez estimada la importancia global, se determina cuántas estructuras deberán ser eliminadas en la iteración actual. Este número se calcula considerando la cantidad de canales existentes y el número máximo de iteraciones especificado, buscando asegurar una reducción gradual y controlada del modelo. Luego, mediante un ordenamiento global de la importancia de las estructuras, se seleccionan aquellas con menor contribución al modelo para ser eliminadas, siempre respetando las restricciones impuestas por el grafo de dependencias, es decir, eliminar o conservar a su vez elementos de la arquitectura que son dependientes entre sí.

Después de realizar la poda física, excluyendo las estructuras seleccionadas, se reconstruye un nuevo modelo podado, que mantiene únicamente las estructuras relevantes restantes y sus pesos.

Este proceso iterativo continúa hasta cumplir con el factor de aceleración deseado o hasta alcanzar el límite de iteraciones.

Finalmente, una vez concluido el procedimiento iterativo, el modelo resultante se re-entrena utilizando el conjunto original de datos. Este re-entrenamiento permite recuperar parcialmente el desempeño predictivo que pudiera haberse afectado como resultado del proceso de poda.

De este modo, el algoritmo general es adaptable a diversos criterios de importancia. Precisamente, la elección adecuada del criterio de importancia constituye uno de los elementos clave del éxito de la poda estructural. En las siguientes secciones se abordan dos propuestas de criterios de importancia basados en técnicas de XAI.

6.2 Criterios de importancia basados en técnicas XAI

El desempeño final de cualquier estrategia de poda estructural depende significativamente del criterio utilizado para estimar la importancia relativa de las estructuras internas del modelo. Tradicionalmente, estos criterios se basan en propiedades directas del modelo, como la magnitud de los pesos o los gradientes. Sin embargo, estos enfoques no siempre reflejan claramente la contribución real de cada estructura al desempeño predictivo del modelo, especialmente en modelos complejos como las redes neuronales convolucionales profundas.

En este trabajo se propone una metodología basada en técnicas de XAI, específicamente en métodos que generan CAMs. La motivación detrás de esta propuesta radica en aprovechar la capacidad de estos métodos para identificar espacialmente las regiones más relevantes de una imagen para la predicción del modelo. Al hacerlo, se obtiene información valiosa acerca de qué filtros específicos generan activaciones que realmente impactan en las decisiones de clasificación.

Concretamente, en esta investigación se explora el uso de EigenCAM y HiResCAM, dos métodos XAI destacados por su precisión y capacidad explicativa, explorada en el CAPÍTULO V.

En las siguientes secciones, se describen formalmente estos dos métodos de estimación de importancia, su fundamentación matemática y la justificación de cómo estos criterios permiten una selección más efectiva de estructuras a eliminar. Además, se aborda específicamente el problema

del desvanecimiento de gradientes mediante una técnica propuesta de normalización por capa, la cual asegura equidad al realizar la poda global.

6.2.1 Importancia basada en EigenCAM

Para determinar la importancia estructural dentro del modelo utilizando la técnica EigenCAM se propone el procedimiento descrito en el Algoritmo 2. EigenCAM se basa en la descomposición SVD aplicada directamente sobre los mapas de activación generados por cada capa convolucional del modelo.

La intuición principal detrás de SVD aplicada a mapas de activación es que el primer vector singular izquierdo $U_{[:,0]}$ apunta en la dirección donde los datos tienen la mayor varianza. Esto implica que, si los mapas de activación están discriminando efectivamente entre diferentes clases, las activaciones correspondientes a distintas clases estarán separadas espacialmente y, por lo tanto, la dirección de mayor varianza capturada por $U_{[:,0]}$ representará esta discriminación. En otras palabras, el vector singular principal captura las características más relevantes que permiten al modelo discriminar efectivamente entre las clases, proporcionando así un criterio robusto de importancia global para cada canal o filtro.

Algoritmo 2: Estimación de importancia S mediante EigenCAM.

Requiere: Modelo podado $\hat{f}(x;W)$, grafo de dependencias DG , límite de mini-lotes N_B , conjunto de datos D .

- | | |
|---|---|
| <p>1: $S = 0$, $S \in R^{ DG }$</p> <p>2: $B_j \subset D$, $j = 1, 2, \dots, N_B$</p> <p>3: Para cada mini-lote B_j:</p> <p style="padding-left: 40px;">4: $M_j \leftarrow f(B_j; W)$</p> <p>5: Por cada grupo de dependencias G en DG:</p> <p style="padding-left: 40px;">6: Por cada capa ℓ en G:</p> | <p>➤ Inicializar el vector de importancia S con ceros. DG es el número de grupos.</p> <p>➤ Obtener N_B mini-lotes del conjunto de datos D.</p> <p>➤ Obtener activaciones de cada capa realizando una pasada hacia adelante del mini-lote.</p> |
|---|---|

7: Aplanar ($M_j^{(\ell)} = \hat{M}_j^{(\ell)} \in \mathbb{R}^{B \times C_\ell \times (H \cdot W)}$)	➤ Redimensionar activaciones de la capa ℓ .
8: Por cada muestra b en B_j :	
9: $U, \Sigma, V \leftarrow \text{SVD}(\hat{M}_{j,b}^{(\ell)})$	➤ Realizar la descomposición en valores singulares.
10: $s_{j,b}^{(\ell,k)} = U_{k,0} $	➤ Calcular la importancia temporal del canal k como la magnitud absoluta del primer vector propio.
11: $s_j^{(\ell,k)} = \frac{1}{ B } \sum_{b=1}^{ B } s_{j,b}^{(\ell,k)}$	➤ Actualizar la importancia promedio por canal en el mini-lote.
12: $S^{(\ell,k)} \leftarrow \frac{1}{N_B} \sum_{j=1}^{N_B} s_j^{(\ell,k)}$	➤ Actualizar acumuladores por canal y normalizar S sobre todos los mini-lotes.
13: $S_G = \frac{1}{ G } \sum_{\ell,k} S^{(\ell,k)}$	➤ Reducción para asignar una importancia global a cada grupo.
14: $S \leftarrow S.append(S_G)$	➤ Añadir importancia del grupo G al vector S .
15: Devolver: S	➤ Devolver vector de importancia S_G .

El procedimiento general del Algoritmo 2 es el siguiente; inicialmente, se considera un conjunto de mini-lotes extraídos del conjunto total de datos con el propósito de reducir posibles sesgos. El riesgo radica en que ciertos mini-lotes podrían sobre-representar o sub-representar algunas clases específicas del conjunto de datos, provocando que el criterio de importancia estimado refleje de manera poco fiable la verdadera relevancia global del filtro. El utilizar varios mini-lotes para calcular el criterio de importancia mitiga este riesgo.

Cada mini-lote se propaga a través del modelo podado, generando mapas de activación en cada capa convolucional. Dichos mapas de activación se transforman en matrices bidimensionales con dimensiones correspondientes al número de filtros (canales) y la dimensión espacial (alto $\times$ ancho).

A continuación, se aplica la descomposición SVD sobre cada una de estas matrices transformadas. La matriz resultante U contiene las direcciones principales de variabilidad de los canales del mapa de activación. En particular, la primera columna de esta matriz ($U_{[:,0]}$) representa

la dirección más relevante de los datos, aquella que captura la mayor proporción posible de varianza.

Para cada canal del modelo, la magnitud absoluta del primer elemento correspondiente del primer vector propio $U_{k,0}$ refleja la relevancia del canal en términos discriminativos. Este valor se interpreta como la contribución relativa del canal a la variabilidad global. Posteriormente, estos valores se promedian sobre todos los ejemplos dentro del mini-lote, obteniendo un valor representativo de la importancia por canal dentro del mini-lote. Para aumentar la robustez, se promedian estos valores nuevamente sobre múltiples mini-lotes, obteniendo finalmente una estimación global confiable para cada canal.

Por último, estos valores de importancia individuales se relacionan con los grupos de dependencia identificados por un grafo de dependencias previo, con el propósito de calificar cada grupo y al podar asegurar una poda estructural coherente y funcional dentro del modelo.

6.2.2 Importancia basada en HiResCAM

Para estimar la importancia de los filtros del modelo mediante la técnica HiResCAM, se propone el procedimiento descrito en el Algoritmo 3. Esta técnica utiliza los gradientes locales de la salida (puntuación de la clase objetivo) con respecto a las activaciones intermedias generadas por cada capa convolucional. A diferencia de otros métodos basados en gradientes promediados globalmente, HiResCAM realiza la multiplicación entre activaciones y gradientes a nivel local, es decir, elemento por elemento, preservando así detalles finos y asegurando que tanto gradientes positivos como negativos sean considerados en la relevancia de cada filtro.

En este trabajo, se adaptó esta estrategia al contexto de poda estructural, en donde interesa una métrica global del aporte de cada filtro. Dado que HiResCAM no genera directamente un peso global como lo hacen otros métodos con Grad-CAM, se propone emplear como métrica la suma promedio absoluta del gradiente local obtenido para cada activación, de ahí se denomina el método como GradAbs. Matemáticamente, el criterio de importancia $s_j^{(\ell,k)}$ para un filtro específico k de la capa ℓ se calcula mediante la norma ℓ_1 del gradiente local, promediado sobre todas las posiciones espaciales, todas las muestras del mini-lote, y sobre varios mini-lotes:

$$s_j^{(\ell,k)} = \frac{1}{N_B |B_j| HW} \sum_{j,b,h,w} \|\nabla y_{j,b,h,w}^{(\ell,k)}\|_1 \quad (6.1)$$

Algoritmo 3: Estimación de importancia S mediante GradAbs.

Requiere: Modelo podado $\hat{f}(x;W)$, grafo de dependencias DG , límite de mini-lotes N_B , conjunto de datos D .

- | | |
|---|---|
| 1: $S = 0$, $S \in R^{ DG }$ | ➤ Inicializar el vector de importancia S con ceros. |
| 2: $B_j \subset D$, $j=1,2,...,N_B$ | ➤ Obtener N_B mini-lotes del conjunto de datos D . |
| 3: Para cada mini-lote B_j: | |
| 4: $y_j^c \leftarrow f(B_j;W)$ | ➤ Realizar propagación hacia adelante y obtener puntuación de la clase objetivo. |
| 5: $\nabla y_j^c \leftarrow \frac{\partial y_j^c}{\partial M}$ | ➤ Calcular gradientes mediante <i>backpropagation</i> del puntaje de la clase objetivo con respecto a las activaciones. |
| 6: Por cada grupo de dependencias G en DG: | |
| 7: Por cada capa ℓ en G: | |
| 8: $s_j^{(\ell,k)} = \frac{1}{N_B B_j HW} \sum_{j,b,h,w} \ \nabla y_{j,b,h,w}^{(\ell,k)}\ _1$ | ➤ Calcular importancia temporal por canal promediando valor absoluto del gradiente sobre espacio, tamaño de mini-lote y número de mini-lotes. |
| 9: $s_j^{(\ell,k)} = \frac{s_j^{(\ell,k)}}{\frac{1}{C_\ell} \sum_k s_j^{(\ell,k)}}$ | ➤ Normalización por la media de los puntajes $s_j^{(\ell,k)}$ de la capa ℓ . |
| 10: $S_G = \frac{1}{ G } \sum_{\ell,k} s^{(\ell,k)}$ | ➤ Reducción para asignar una importancia global a cada grupo. |
| 11: $S \leftarrow S.append(S_G)$ | ➤ Añadir importancia del grupo G al vector S . |
| 12: Devolver: S | ➤ Devolver vector de importancia S_G . |
-

Este procedimiento asegura que aquellos filtros que exhiben mayores gradientes, ya sean positivos o negativos, es decir, una mayor sensibilidad local de la clase objetivo frente a cambios en las activaciones, sean considerados más importantes para la predicción del modelo.

Si bien este enfoque presenta similitudes conceptuales con técnicas como SNIP [81], GraSP [82] y SynFlow [83] en cuanto al uso de gradientes para medir importancia, cabe destacar que estos trabajos utilizan gradientes de la función de pérdida respecto a los parámetros del modelo. En cambio, el método aquí propuesto emplea gradientes de la puntuación para una clase con respecto a las activaciones intermedias. Ambas perspectivas comparten la premisa de que una mayor magnitud de gradientes indica mayor importancia, pero su interpretación difiere en que el gradiente respecto a activaciones destaca la relevancia local y detallada de patrones específicos aprendidos por filtros individuales, mientras que el gradiente respecto a parámetros refleja una contribución global al desempeño del modelo. La ventaja de la propuesta basada en HiResCAM radica en que permite preservar características locales importantes y detalladas que otros métodos más globales podrían ignorar.

6.2.2.1 Normalización por capa del criterio de importancia

Una limitante detectada al utilizar directamente el criterio basado en HiResCAM es el fenómeno del desvanecimiento de gradientes. Este fenómeno provoca que las capas más cercanas a la entrada generen valores de gradiente sustancialmente menores que las capas más profundas, no debido a una menor importancia sino como consecuencia natural del proceso de retropropagación en redes profundas.

Este problema introduce una desventaja para las capas iniciales al considerar la poda globalmente, resultando en una eliminación no deseada de filtros potencialmente útiles que presentan magnitudes numéricas menores, pero relevancia conceptual comparable.

Como solución a esta problemática, se propone una normalización por capa. En concreto, cada puntaje $s_j^{(\ell,k)}$ de importancia es dividido por la media de los puntajes obtenidos en su misma capa:

$$s_j^{(\ell,k)} = \frac{s_j^{(\ell,k)}}{\frac{1}{C_\ell} \sum_k s_j^{(\ell,k)}} \quad (6.2)$$

Esto garantiza que la media de los puntajes en cada capa sea igual a 1, eliminando sesgos numéricos debidos al desvanecimiento del gradiente. De esta manera, el ranking global de poda es equitativo entre capas, respetando la relevancia relativa dentro de cada capa y asegurando que ninguna capa domine el ranking únicamente por razones numéricas.

Con la definición de los criterios de importancia basados en técnicas XAI y la integración de estos con el algoritmo de poda estructural iterativa, se cuenta ahora con una metodología para la reducción eficiente de redes neuronales convolucionales. La propuesta combina la interpretabilidad que los métodos como EigenCAM y HiResCAM con un enfoque iterativo que conserva una estructura funcional del modelo y permite ajustar el factor de aceleración que se desee.

En el siguiente capítulo se evaluará esta metodología, contrastando su desempeño con diferentes métodos de poda establecidos y analizando su impacto tanto en exactitud como en eficiencia computacional bajo distintos experimentos.

CAPÍTULO VII. RESULTADOS Y CONCLUSIONES

Este capítulo presenta los resultados de los experimentos obtenidos al aplicar la metodología propuesta de poda estructural basada en XAI, así como la comparación con métodos de poda convencionales.

Se analizan distintos factores de aceleración, arquitecturas de red y configuraciones de cálculo de importancia, incluyendo variaciones en el número de mini-lotes empleados, con el objetivo de determinar el compromiso óptimo entre eficiencia y exactitud. Además, se discuten los hallazgos, las limitaciones identificadas y el posible trabajo futuro.

7.1 Resultados globales de la poda estructural

7.1.1 Métricas y umbrales de calidad

La interpretación de los resultados de poda estructural requiere un marco común para comparar la precisión de los modelos podados con respecto a sus versiones originales.

El desempeño de las predicciones se reporta mediante el *accuracy* Top-1 en los conjuntos de prueba de CIFAR-10 y CIFAR-100, tanto después del reentrenamiento como en modalidad *one-shot*. Para facilitar la comparación con el modelo de referencia, se utiliza la diferencia en puntos porcentuales (ΔAcc).

$$\Delta Acc = Acc_{podado} - Acc_{original} \quad (7.1)$$

Con base en este indicador se establecen umbrales que definen diferentes zonas de tolerancia:

- Zona estricta: pérdida máxima de 1 punto porcentual de *accuracy*.
- Zona tolerante: pérdida máxima de 2 puntos porcentuales de *accuracy*.
- Colapso: pérdidas superiores a 5 puntos o *accuracy* cercana a cero.

También se considera la eficiencia computacional, que en este capítulo se analiza desde dos perspectivas, utilizando las métricas definidas en el CAPÍTULO III. Primero con el porcentaje de reducción de FLOPs, que se asocia a la latencia y al consumo energético en inferencia, y en segundo lugar con el porcentaje de reducción de parámetros, relacionada con el tamaño del modelo

y el uso de memoria. Para sintetizar ambas, se usa además al *speed-up* teórico, entendido como el factor de aceleración alcanzado con respecto al modelo original.

7.1.2 FLOPs vs accuracy

El análisis del desempeño en función de la reducción de FLOPs permite evaluar la relación entre eficiencia computacional y precisión, lo que resulta especialmente relevante cuando el objetivo de la compresión es disminuir la latencia de inferencia o el consumo energético. Las curvas de *trade-off* obtenidas muestran que los métodos tienen diferentes resultados en su capacidad de sostener el *accuracy* conforme se incrementa el nivel de poda.

En ResNet-50, la arquitectura más resiliente, se observa que varios criterios mantienen un rendimiento estable hasta reducciones cercanas al 90% de FLOPs (Figura 7.1). En este escenario, GradAbs logra competir de manera directa con aproximaciones más establecidas como la Aproximación de Taylor y la Regularización creciente. No obstante, al superar este umbral, GradAbs tiende a mostrar caídas más pronunciadas, mientras que la Regularización creciente logra mantener niveles de exactitud más altos incluso en la zona extrema.

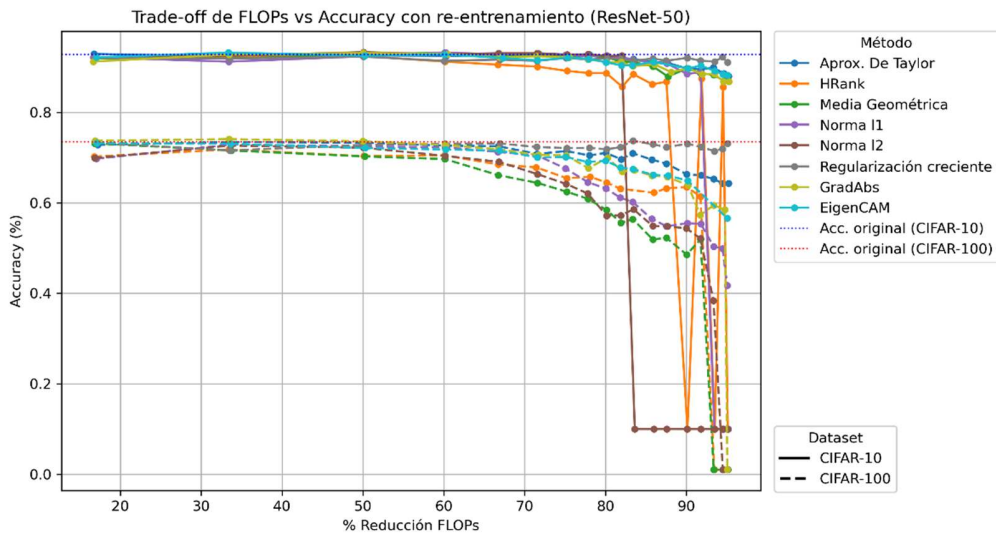


Figura 7.1. Relación entre reducción de FLOPs y precisión con re-entrenamiento para ResNet-50.

En ResNet-18 (Figura 7.2) y DenseNet-121 (Figura 7.3) las diferencias son más evidentes. Aunque en reducciones moderadas, entre 70 y 80%, GradAbs conserva valores de *accuracy*

competitivos, a partir de niveles de compresión más agresivos el método pierde robustez frente a Regularización creciente, que muestra curvas más planas. En el caso de ResNet-18, se muestran colapsos abruptos cuando se buscan reducciones superiores al 90%.

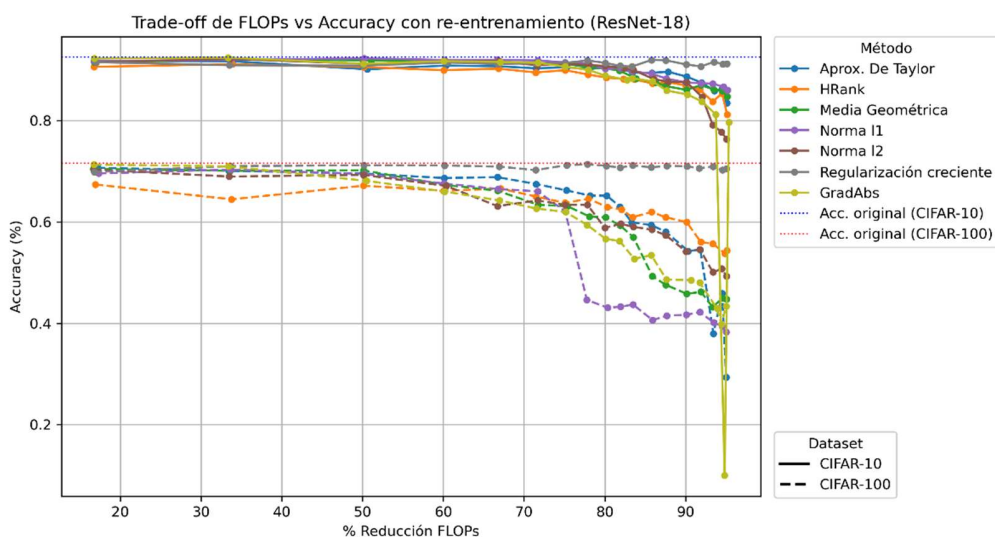


Figura 7.2. Relación entre reducción de FLOPs y precisión con re-entrenamiento para ResNet-18.

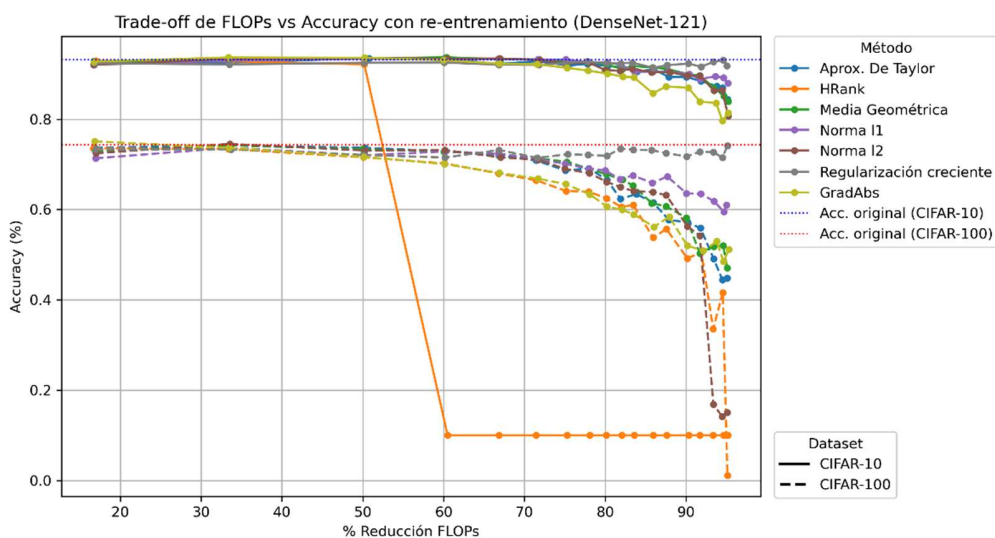


Figura 7.3. Relación entre reducción de FLOPs y precisión con re-entrenamiento para DenseNet-121.

La situación más crítica se presenta en VGG19-BN (Figura 7.4), donde los modelos muestran una marcada vulnerabilidad a la poda. En este caso, ninguna técnica alcanza reducciones agresivas

de FLOPs sin pérdidas severas de exactitud. GradAbs, al igual que la mayoría de los métodos, experimenta caídas abruptas, confirmando que arquitecturas sin atajos residuales son considerablemente menos tolerantes a la compresión estructural.

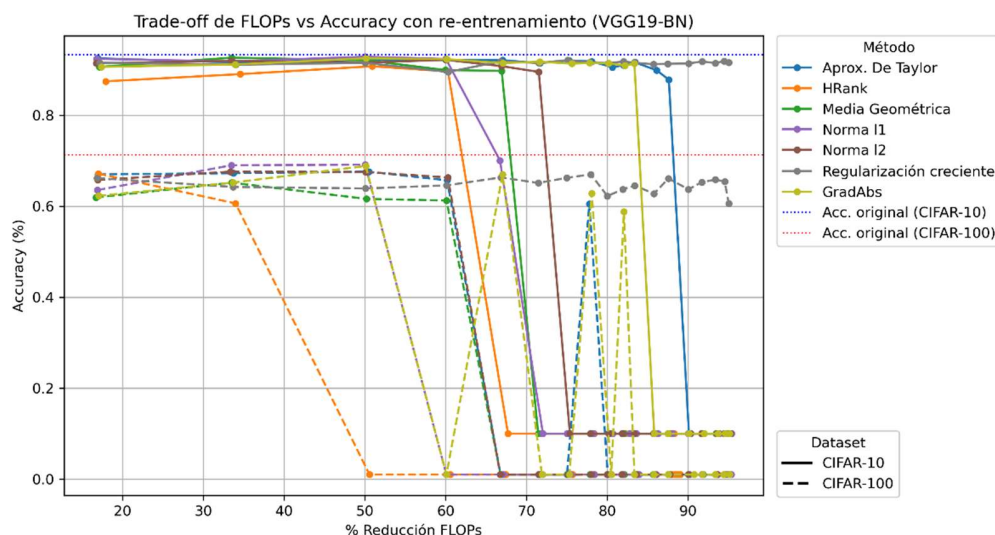


Figura 7.4. Relación entre reducción de FLOPs y precisión con re-entrenamiento para VGG19-BN.

En términos generales, puede concluirse que GradAbs es una alternativa viable en escenarios donde se persigue una reducción moderada a alta de FLOPs (entre 70 al 85%) en arquitecturas residuales o densas, pero resulta menos competitivo cuando se empuja la poda hacia el límite. Para esos casos, el método de Regularización creciente se posiciona como la opción más robusta, ya que logra preservar la *accuracy* incluso en niveles de compresión extrema.

7.1.3 Parámetros vs accuracy

El análisis en función de la reducción de parámetros muestra la capacidad de cada criterio para disminuir el tamaño del modelo y, por lo tanto, su huella en memoria y almacenamiento. A diferencia de la reducción de FLOPs, este eje revela que algunos métodos tienden a priorizar la poda de canales en capas profundas, lo que conlleva una reducción significativa del número de parámetros sin un impacto equivalente en la aceleración teórica.

Los resultados muestran que GradAbs es particularmente efectivo en este aspecto. En ResNet-50, logra reducciones de parámetros superiores al 95% en escenarios con pérdidas de exactitud inferiores a un punto porcentual, superando a la mayoría de los métodos de referencia (Figura 7.5).

Este comportamiento se repite en DenseNet-121, donde las reducciones alcanzan valores cercanos al 92% (Figura 7.6). En ambos casos, las curvas revelan que el método permite alcanzar niveles de compresión muy altos sin comprometer de manera inmediata la exactitud, lo que confirma su inclinación hacia arquitecturas más compactas.

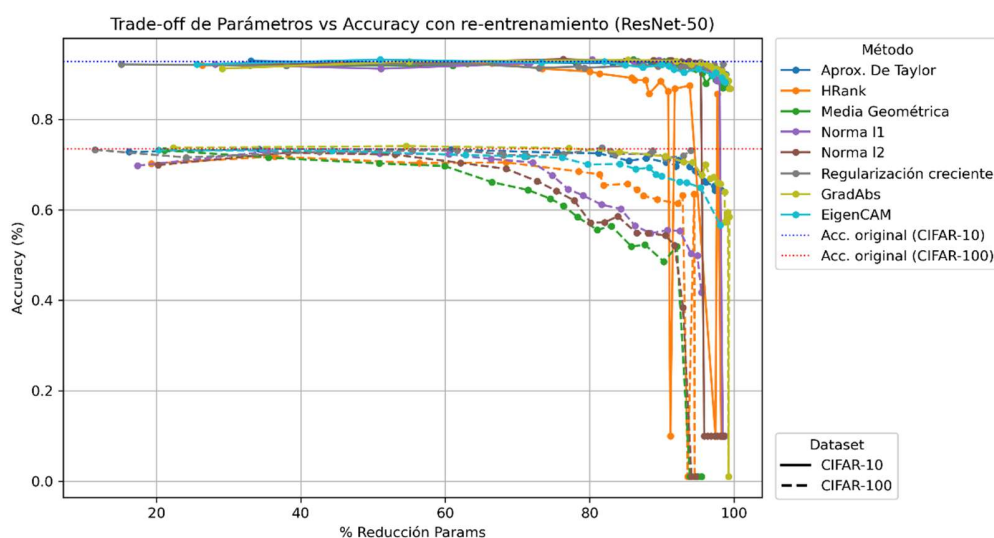


Figura 7.5. Relación entre reducción de parámetros y precisión con re-entrenamiento para ResNet-50.

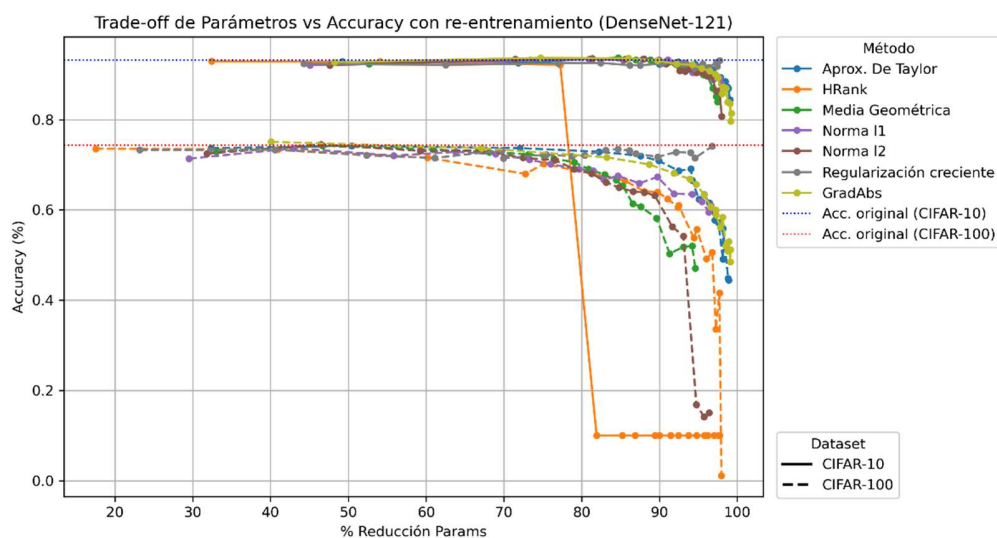


Figura 7.6. Relación entre reducción de parámetros y precisión con re-entrenamiento para DenseNet-121.

En contraste, con ResNet-18 (Figura 7.7) y VGG19-BN (Figura 7.8) la situación es distinta. Si bien GradAbs mantiene un lugar competitivo, los niveles de reducción obtenidos no siempre se

traducen en ganancias de exactitud. En estas arquitecturas menos resilientes a la poda, el incremento de compresión más allá del 85% de parámetros suele acompañarse de caídas abruptas en el *accuracy*, un patrón consistente también en el resto de métodos de poda.

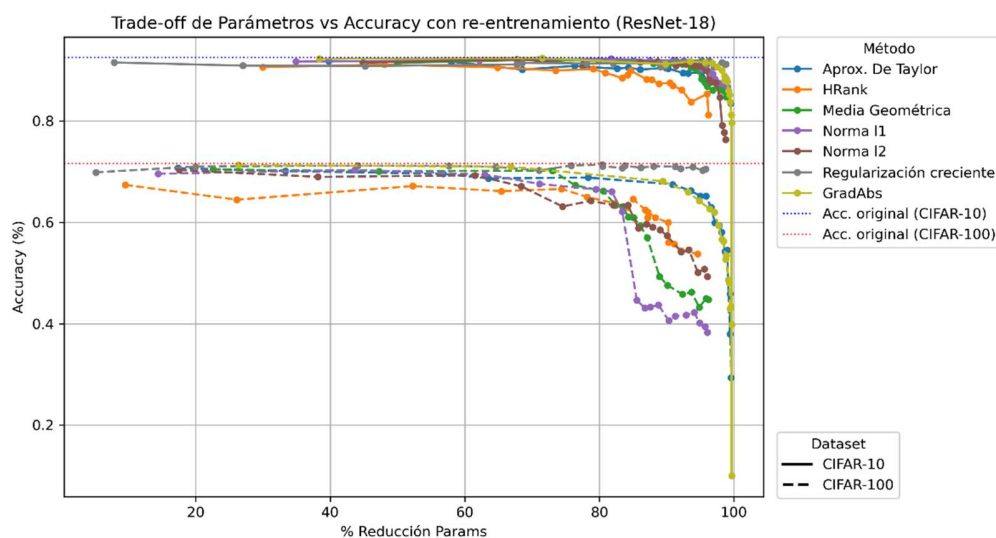


Figura 7.7. Relación entre reducción de parámetros y precisión con re-entrenamiento para ResNet-18.

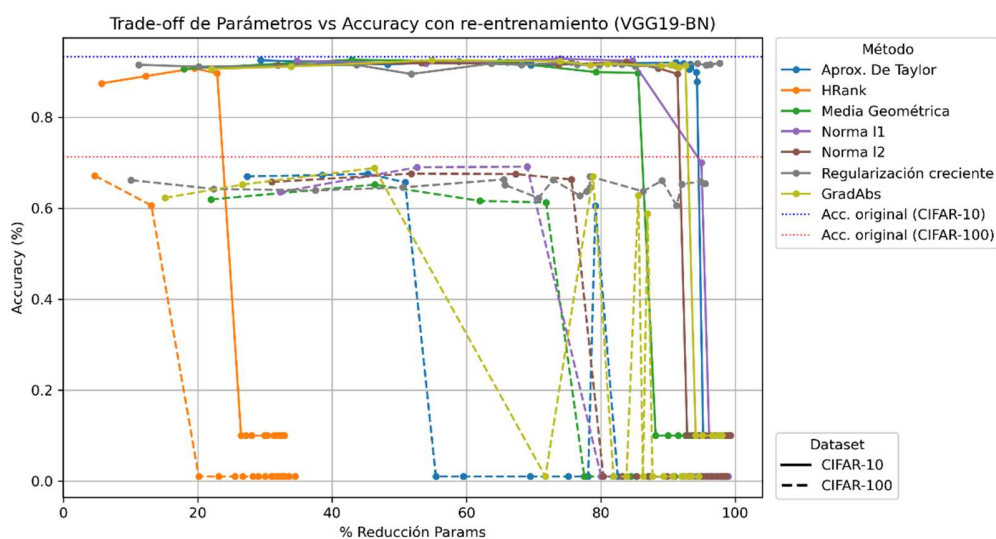


Figura 7.8. Relación entre reducción de parámetros y precisión con re-entrenamiento para VGG19-BN.

La comparación entre reducción de FLOPs y reducción de parámetros permite identificar que GradAbs, para un mismo nivel de tolerancia en pérdida de precisión, se eliminan más parámetros que FLOPs. En términos prácticos, este comportamiento lo convierte en un criterio preferible

cuando el objetivo principal es disminuir el tamaño del modelo, por ejemplo, en escenarios donde la restricción de memoria es más crítica que la latencia.

7.2 Comparativa y sensibilidad de los criterios de importancia

La eficacia de un criterio de poda no depende únicamente de su formulación matemática, sino también de su costo computacional, su estabilidad frente a distintas arquitecturas y la manera en que responde a las condiciones de entrenamiento. En esta sección se comparan los resultados obtenidos con GradAbs frente a otros enfoques, incluyendo un estudio piloto con EigenCAM y un análisis de sensibilidad respecto al número de mini-lotes utilizados para estimar importancias.

7.2.1 GradAbs vs EigenCAM

La comparación entre GradAbs y EigenCAM se llevó a cabo sobre ResNet-50. Los resultados indican que ambos criterios producen curvas de porcentaje de reducción de FLOPs vs. *accuracy* (Figura 7.1) de forma muy similar en la zona de compresión moderada, alrededor del 70 al 85%, lo que sugiere que GradAbs captura patrones de importancia equiparables a los derivados de SVD.

La principal diferencia radica en el costo computacional, mientras que EigenCAM requiere del cálculo de SVD, que implica operaciones adicionales de factorización, GradAbs se obtiene directamente a partir de los gradientes, lo que lo convierte en una alternativa más escalable y práctica para experimentos masivos o arquitecturas de mayor tamaño, aunque EigenCAM cuenta con mejor interpretabilidad visual, GradAbs logra un desempeño comparable con una complejidad significativamente menor.

7.2.2 GradAbs vs métodos de poda de referencia

La comparación entre GradAbs y los métodos de poda de referencia revela diferencias claras en función del objetivo de compresión, FLOPs o parámetros, y de la agresividad de la poda aplicada. Bajo la restricción de calidad $\Delta Acc \geq -1$, los resultados muestran que la Regularización creciente es el método más eficaz para maximizar la reducción de FLOPs y, en consecuencia, el factor de aceleración (Figura 7.1-Figura 7.8). Sin embargo, este resultado debe interpretarse con cuidado, a diferencia del resto de enfoques, la Regularización creciente no se aplica directamente sobre el modelo entrenado, sino que requiere un entrenamiento adicional previo, durante el cual se incorpora una regularización en la función de pérdida que favorece la dispersión de los pesos y

facilita la poda posterior. Este procedimiento otorga una ventaja inherente a este método, pero también lo hace más costoso y menos comparable en igualdad de condiciones con los demás criterios.

Aún así, al analizar los resultados orientados a la reducción de FLOPs, la a Tabla 7.1, con el promedio del mayor porcentaje de reducción de FLOPs cuando $\Delta Acc \geq -1$ por método, muestra que la Regularización creciente alcanza un valor promedio de 70.77%, muy por encima del resto de métodos. GradAbs, en cambio, obtiene 50.16%, situándose junto a métodos como la aproximación de Taylor y norma ℓ_1 . Esta diferencia refleja la ventaja que otorga el entrenamiento regularizado previo, ya que genera modelos dispersos que favorecen a la poda.

Tabla 7.1. Promedio de los máximos % de reducción de FLOPs de cada método logrando un $\Delta Acc \geq -1$.

Método	Máximo % de reducción de FLOPs con restricción de $\Delta Acc \geq -1$									Promedio de máximo % de reducción de FLOPs
	ResNet-50		ResNet-18		DenseNet-121		VGG19-BN			
	CIFAR-10	CIFAR-100	CIFAR-10	CIFAR-100	CIFAR-10	CIFAR-100	CIFAR-10	CIFAR-100		
Regularización creciente	94.4800%	95.100%	93.5100%	93.3900%	94.59%	95.12%	0%	0%	70.7738%	
Norma l1	82.0900%	33.700%	71.7100%	0%	80.2800%	33.55%	60.03%	0%	45.1700%	
Norma l2	82.0100%	33.6500%	66.9400%	0%	77.9700%	33.54%	0%	0%	36.7638%	
Media Geométrica	80.1900%	16.8300%	60.1200%	0%	80.1200%	0%	33.53%	0%	33.8488%	
Aprox. De Taylor	83.4200%	66.7400%	33.9300%	17.1900%	77.7900%	50.21%	16.92%	0%	43.2750%	
GradAbs	75.5300%	60.200%	67.04%	33.4000%	71.56%	33.45%	60.16%	0%	50.1675%	
HRank	66.48%	0%	0%	0%	33.700%	33.42%	0%	0%	16.7000%	

De manera similar, en la Tabla 7.2 se muestra el análisis del promedio del mayor porcentaje de reducción de parámetros cuando $\Delta Acc \geq -1$. En este caso se observa que GradAbs alcanza un promedio de alrededor del 72.28% y se encuentra ligeramente debajo del 72.39% alcanzado por el método de Regularización creciente, mientras que el resto de los métodos cuentan con un peor desempeño. Esto evidencia que, aunque la Regularización creciente sigue liderando, GradAbs ofrece una alternativa competitiva cuando se busca reducir la cantidad de parámetros, además de que no requiere un entrenamiento adicional especializado.

Tabla VII.2. Promedio de los máximos % de reducción de parámetros de cada método logrando un $\Delta Acc \geq -1$.

Método	Máximo % de reducción de parámetros con restricción de $\Delta Acc \geq -1$								Promedio de máximo % de reducción de parámetros
	ResNet-50		ResNet-18		DenseNet-121		VGG19-BN		
	CIFAR-10	CIFAR-100	CIFAR-10	CIFAR-100	CIFAR-10	CIFAR-100	CIFAR-10	CIFAR-100	
Regularización creciente	98.49%	94.03%	98.22%	93.92%	97.74%	96.79%	0%	0%	72.3988%
Norma l1	95.75%	35.37%	92.52%	0%	93.0900%	43.03%	84.71%	0%	55.5588%
Norma l2	95.41%	38.24%	88.68%	0%	91.93%	46.51%	0%	0%	45.0963%
Media Geométrica	93.64%	21.14%	85.04%	0%	92.18%	60.92%	42.97%	0%	49.4863%
Aprox. De Taylor	95.01%	81.24%	57.97%	17.37%	94.60%	72.08%	29.37%	0%	55.9550%
GradAbs	96.27%	84.16%	95.77%	66.85%	94.20%	67.12%	73.94%	0%	72.2888%
HRank	66.48%	0%	0%	0%	54.08%	40.95%	0%	0%	20.1888%

7.2.3 Impacto del número de mini-lotes en GradAbs

El criterio GradAbs se fundamenta en calcular la importancia de los filtros a partir de la magnitud media de sus gradientes. Una característica de este procedimiento es que no se evalúa el gradiente sobre el conjunto de datos completo, sino sobre un subconjunto de mini-lotes N_B , lo que introduce un hiperparámetro adicional que podría influir en la estabilidad y el desempeño final del método.

Para analizar esta sensibilidad, se realizaron experimentos variando el número de mini-lotes considerados $N \in \{1, 3, 5, 10, 20\}$ en las arquitecturas ResNet-50 con los conjuntos de datos CIFAR-10 y CIFAR-100. Las gráficas asociadas a esta sección muestran las curvas de *accuracy* frente a reducción de FLOPs, así como la distribución de filtros podados por capa en diferentes niveles de compresión.

La Figura 7.9 y Figura 7.10 muestran que el número de mini-lotes empleados tiene un pequeño efecto en el comportamiento global de GradAbs. En CIFAR-100, las curvas prácticamente se superponen hasta reducciones del 80% de FLOPs, y solo en escenarios extremos ($> 90\%$) se observa un colapso más abrupto cuando se usan pocos lotes, por ejemplo 5. En CIFAR-10, se aprecia que con cualquier configuración de mini-lotes se alcanza un pico de *accuracy* que supera al modelo original, lo cual refuerza la hipótesis de que la poda induce un efecto de regularización. La diferencia entre configuraciones es marginal, las variaciones entre 1 y 20 lotes se mantienen dentro de 0.5 puntos porcentuales de *accuracy* en la mayor parte del rango. Estos resultados sugieren que GradAbs no depende críticamente del número de mini-lotes para preservar el desempeño, ya que incluso estimaciones con un único lote ofrecen resultados comparables a las de configuraciones más costosas.

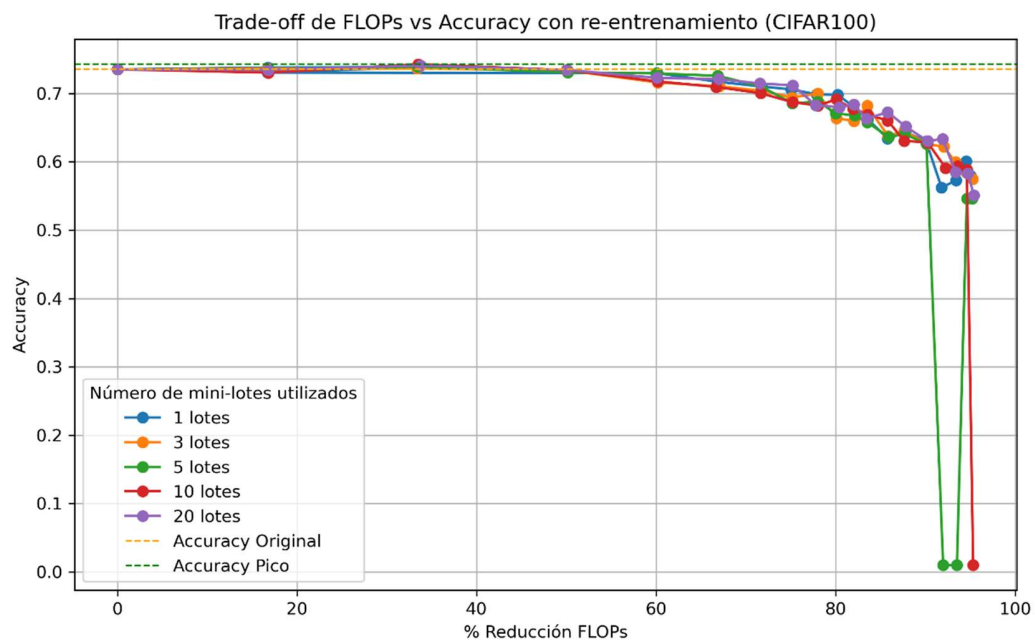


Figura 7.9. Trade-off FLOPs vs accuracy en CIFAR-100 con variación de mini-lotes.

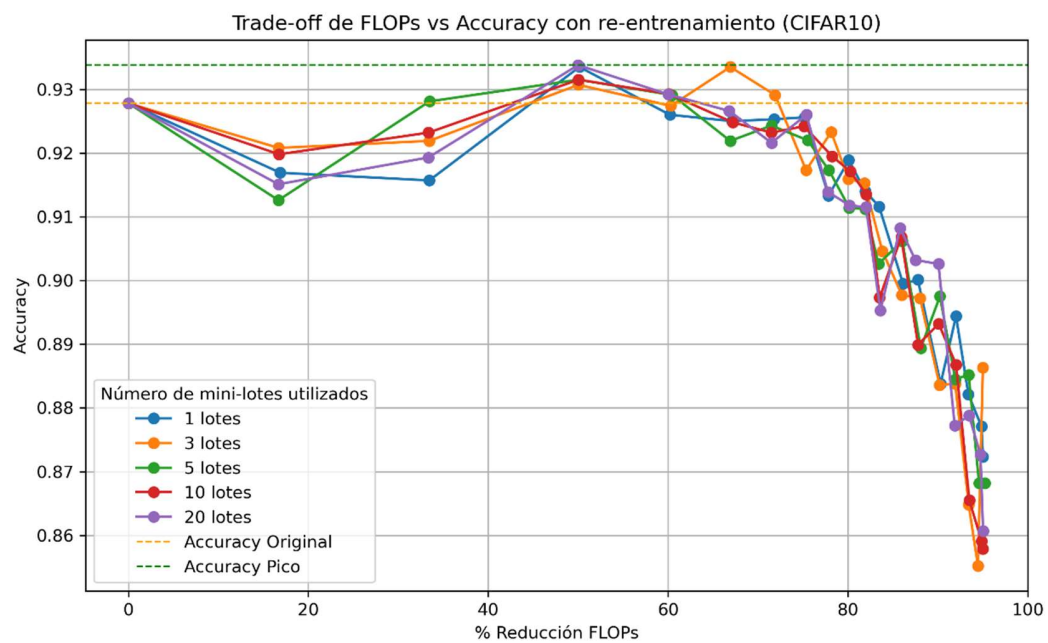


Figura 7.10. Trade-off FLOPs vs accuracy en CIFAR-10 con variación de mini-lotes.

El análisis se complementa con la Figura 7.11 y la Figura 7.12, que ilustran el número de filtros eliminados en cada capa de ResNet-50 al alcanzar factores de aceleración de $1.5\times$ y $20\times$. De nuevo, la variabilidad entre configuraciones es mínima, se puede observar que las barras de 1, 3, 5, 10 y 20 lotes prácticamente se superponen en la mayoría de las capas. Esto indica que la estimación de importancia a partir de pocos mini-lotes no solo conserva la *accuracy* global, sino que también produce una poda estructural consistente.

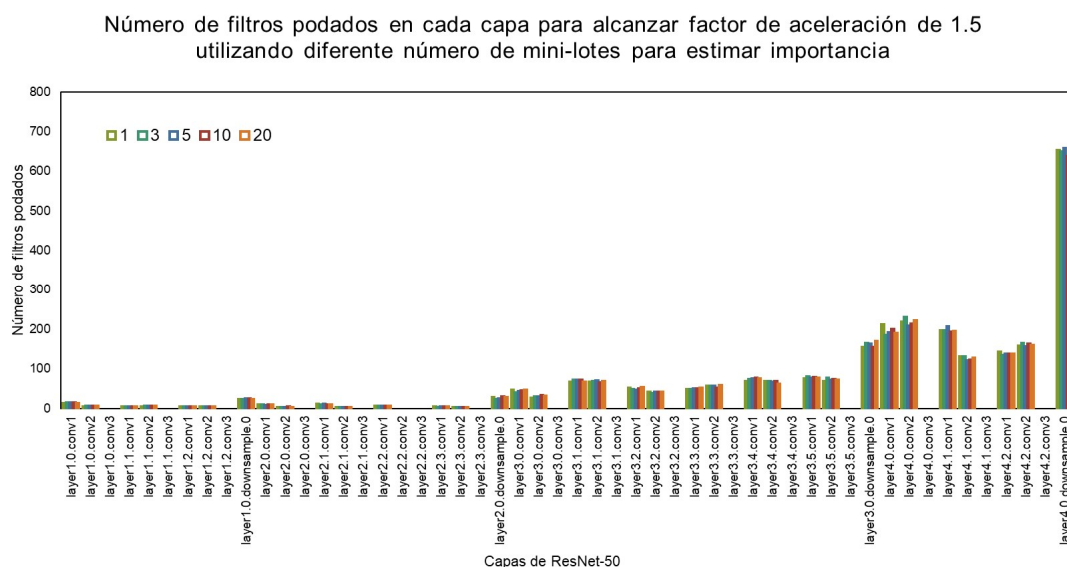


Figura 7.11. Filtros podados por capa en ResNet-50 con factor de aceleración $1.5\times$ y variación de mini-lotes.

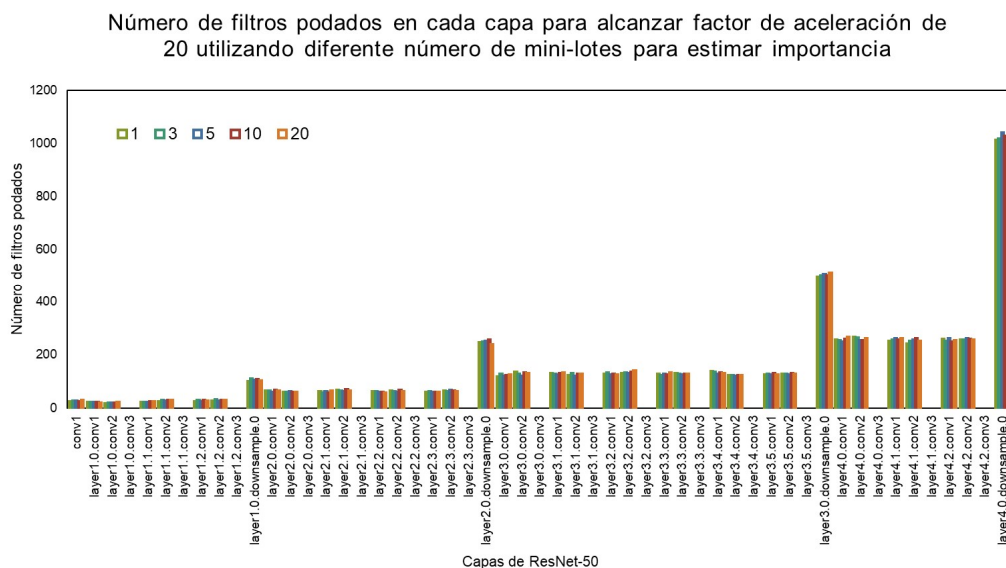


Figura 7.12. Filtros podados por capa en ResNet-50 con factor de aceleración $20\times$ y variación de mini-lotes.

Con esto se puede confirmar que GradAbs ofrece una estimación de importancia suficientemente robusta incluso utilizando pocos mini-lotes, sin comprometer ni el desempeño ni la coherencia estructural de la poda.

7.3 Efecto de la arquitectura y del dataset

La tolerancia de un modelo a la poda estructural no depende únicamente del criterio empleado, sino también de la arquitectura y de la dificultad del conjunto de datos. En este sentido, los resultados muestran un contraste entre arquitecturas profundas y densas, que exhiben una mayor resiliencia, y arquitecturas más ligeras o menos redundantes, donde la compresión se reduce de forma considerable.

7.3.1 Arquitecturas resilientes

En arquitecturas de gran profundidad y con redundancia como ResNet-50 y DenseNet-121, algunos métodos alcanzaron desempeños que incluso superaron el *accuracy* original tras la poda y el re-entrenamiento. Este fenómeno evidencia que, al eliminar filtros poco relevantes, no solo se reduce el costo computacional, sino que también puede mejorar la capacidad de generalización del modelo.

La Tabla 7.3 presenta los casos en los que se superó el *accuracy* inicial. Se observa que, tanto en CIFAR-10 como en CIFAR-100, GradAbs logra podas agresivas en parámetros, entre el 22% y 92%, y FLOPs entre 16% y 60%, mientras mantiene o mejora la precisión. Además, se observa que GradAbs es el único método de poda que logra generar modelos que sobrepasan el desempeño original en ambas arquitecturas de DNNs y con los dos conjuntos de datos. Esto muestra que GradAbs es capaz de explotar la redundancia en arquitecturas complejas de manera consistente.

Tabla 7.3. Casos que superan el *accuracy* original.

Arquitectura	Conjunto de datos	Factor de aceleración alcanzado	Método de poda	% Reducción de Parámetros	% Reducción de FLOPs	Accuracy inicial	Accuracy luego de la poda y re-entrenamiento	Δ Acc
ResNet-50	CIFAR-10	1.2	Aprox. De Taylor	33.08%	16.77%	92.78%	92.94%	0.16%
		3.5		86.71%	71.49%		92.83%	0.05%
		4.5		90.31%	77.99%		92.80%	0.02%
		2.5	Media Geométrica	86.06%	60.13%		93.27%	0.49%
		3.5		91.00%	71.72%		93.03%	0.25%
		2.5	Norma l1	80.35%	60.23%		93.25%	0.47%
		3		86.09%	66.78%		92.84%	0.06%
		3.5		89.61%	71.56%		93.10%	0.32%
		2	Norma l2	76.36%	50.07%		93.34%	0.56%
		2.5		84.92%	60.18%		92.87%	0.09%
		3		88.87%	66.79%		93.10%	0.32%
		3.5		91.30%	71.54%		93.11%	0.33%
		4		92.73%	75.25%		92.81%	0.03%
		4.5		93.79%	77.86%		92.89%	0.11%
		1.5	GradAbs	55.03%	33.48%		92.81%	0.03%
		2		85.32%	50.03%		93.15%	0.37%
		2.5		92.00%	60.46%		92.91%	0.13%
		1.5	EigenCAM	51.02%	33.39%		93.24%	0.46%
		2.5		82.06%	60.08%		92.83%	0.05%
	CIFAR-100	6	Regularización creciente	81.73%	83.38%	73.52%	73.75%	0.23%
		1.2	GradAbs	22.33%	16.89%		73.74%	0.22%
		1.5		54.60%	33.46%		74.11%	0.59%
		2		77.15%	50.01%		73.63%	0.11%
DenseNet-121	CIFAR-10	2	Aprox. De Taylor	81.32%	50.77%	93.17%	93.51%	0.34%
		2.5		87.92%	60.02%		93.39%	0.22%
		1.5	Media Geométrica	73.38%	33.53%		93.18%	0.01%
		2		81.06%	50.37%		93.47%	0.30%
		2.5		84.69%	60.39%		93.72%	0.55%
		3		86.94%	67.04%		93.28%	0.11%
		3.5		88.83%	71.82%		93.27%	0.10%
		2	Norma l1	80.97%	50.23%		93.40%	0.23%
		3		87.89%	66.79%		93.38%	0.21%
		3.5		89.91%	71.71%		93.23%	0.06%
		4		91.08%	75.11%		93.22%	0.05%
		1.5	Norma l2	71.46%	33.36%		93.38%	0.21%
		2		81.36%	50.25%		93.55%	0.38%
		2.5		85.61%	60.49%		93.40%	0.23%
		3		88.02%	66.92%		93.45%	0.28%
		1.5	GradAbs	74.73%	33.40%		93.69%	0.52%
		2		85.99%	50.22%		93.60%	0.43%
	CIFAR-100	1.5	Norma l2	46.51%	33.54%	74.36%	74.52%	0.16%
		1.2	GradAbs	40.08%	16.83%		75.08%	0.72%

7.3.2 Arquitecturas frágiles

En contraste, arquitecturas más ligeras como ResNet-18 y VGG19-BN muestran una menor tolerancia a la poda. Ningún método logró superar el desempeño original, lo que sugiere que estas arquitecturas cuentan con menos redundancia y son más sensibles a la eliminación de filtros.

No obstante, aunque GradAbs no alcanza los factores de aceleración más altos como los logrados mediante Regularización creciente, su comportamiento se caracteriza por ser consistente y robusto. En arquitecturas resilientes, logra frecuentemente igualar o incluso superar el desempeño original, mientras que en arquitecturas frágiles, donde ningún método consigue

sobrepasar al original, GradAbs se mantiene de forma sistemática entre los modelos con mejor rendimiento. La Tabla 7.4 muestra que aun sin liderar en reducción de FLOPs, GradAbs evita los colapsos drásticos observados en métodos como HRank y asegura modelos que, aunque no siempre superan al original, sí se posicionan de manera recurrente en el conjunto de mejores desempeños alcanzados.

Tabla 7.4. Top-3 de mejores desempeños en modelos entrenados con CIFAR-10 y CIFAR-100 de las arquitecturas ResNet-18 y VGG19-BN.

Arquitectura	Conjunto de datos	Factor de aceleración alcanzado	Método de poda	% Reducción de Paráms.	% Reducción de FLOPs	Accuracy inicial	Accuracy luego de la poda y re-entrenamiento
ResNet-18	CIFAR-10	1.5	GradAbs	71.52%	33.36%	92.59%	92.44%
		2	Norma l1	81.76%	50.22%		92.29%
		1.2	GradAbs	38.38%	16.79%		92.28%
	CIFAR-100	4.5	Regularización creciente	80.46%	77.86%	71.58%	71.38%
		1.2	GradAbs	26.35%	16.76%		71.33%
		4	Regularización creciente	75.84%	75.29%		71.23%
VGG19-BN	CIFAR-10	2	Norma l1	73.98%	50.02%	93.28%	92.88%
		1.5	Media Geométrica	42.97%	33.53%		92.66%
		2	GradAbs	54.85%	50.15%		92.53%
	CIFAR-100	2	Norma l1	69.03%	50.03%	71.28%	69.14%
		1.5	Norma l1	52.65%	33.47%		69.00%
		2	GradAbs	46.30%	50.10%		68.85%

7.3.3 Impacto del conjunto de datos.

El conjunto de datos también influye significativamente en los resultados. Con CIFAR-10, los modelos podados presentan mayor margen para sostener o incluso superar la precisión original, lo que se atribuye a la menor complejidad de las clases y la abundancia relativa de ejemplos. En cambio, con CIFAR-100, las diferencias entre métodos se amplifican y los límites de compresión segura aparecen antes, reflejando la mayor dificultad del problema de clasificación.

7.4 Conclusiones

Del trabajo realizado se llegan a un conjunto de conclusiones sobre la compresión de redes neuronales profundas mediante poda estructural guiada por técnicas de explicabilidad.

En primer lugar, se confirma que el re-entrenamiento posterior a la poda es imprescindible. Todas las técnicas analizadas deterioran inicialmente la exactitud al eliminar filtros, sin embargo, con un protocolo de re-entrenamiento adecuado como el que se sigue en esta tesis (poda progresiva,

scheduler con reducción en *plateau* y *early stopping*) es posible recuperar e incluso superar el desempeño original en varios casos.

En segundo lugar, los resultados muestran que no todas las arquitecturas toleran la compresión por igual. Modelos profundos y densos como ResNet-50 y DenseNet-121 demostraron una notable resiliencia, permitiendo reducciones de hasta 90% de FLOPs sin comprometer la precisión. En cambio, arquitecturas más ligeras como ResNet-18 y VGG19-BN resultaron frágiles, ya que ninguna técnica logró reducciones agresivas sin pérdidas severas, lo que refleja la menor redundancia en la estructura de estos modelos.

Asimismo, el análisis evidencia que el conjunto de datos utilizado es un factor determinante. En CIFAR-10 se observa mayor margen para mantener o incluso mejorar el *accuracy* tras la poda, mientras que en CIFAR-100 los límites de compresión aparecen más temprano y las diferencias entre métodos se amplifican. Esto sugiere que la complejidad de la tarea debe guiar los niveles de poda aplicados en la práctica.

Otro hallazgo relevante es la identificación de un punto óptimo de compresión, más allá del cual cada reducción adicional incrementa el costo de re-entrenamiento y provoca caídas de exactitud. La compresión extrema, si bien es posible en ciertos escenarios, deja de ser ventajosa cuando se valoran tanto exactitud como el costo de re-entrenamiento.

El criterio propuesto en esta tesis, denominado GradAbs y derivado de HiResCAM, se validó como una alternativa competitiva y robusta en rangos de compresión moderada. Al basarse en el promedio de valores absolutos del gradiente local, con normalización por capa para mitigar el desvanecimiento de gradientes, GradAbs logra desempeños estables, comparables a métodos como Aproximación de Taylor o Regularización creciente. Si bien en compresiones extremas esta última conserva mejor la precisión, GradAbs evita colapsos en arquitecturas frágiles y mantiene modelos siempre competentes.

Los experimentos también revelaron casos en que la poda más re-entrenamiento conduce a un desempeño superior al original, en particular en CIFAR-10 con DenseNet-121 y ResNet-50, a factores de aceleración de entre $1.5\times$ y $3.5\times$. Esto sugiere que una poda moderada puede funcionar como un mecanismo de regularización, reduciendo el sobreajuste y mejorando la generalización.

En términos prácticos, los resultados permiten formular las siguientes recomendaciones:

- Para máxima compresión con mínima pérdida, la regularización creciente sigue siendo la opción más potente, aunque requiere un entrenamiento adicional previo.
- Para compresión moderada, estable y de bajo costo, GradAbs es una buena alternativa como un criterio seguro y consistente.
- En escenarios donde es necesaria una fuerte restricción de memoria, GradAbs resulta especialmente ventajoso por su mayor reducción de parámetros en relación a FLOPs.
- Es preferible utilizar arquitecturas con conexiones residuales o densas como ResNet-50 y DenseNet-121 cuando se planean compresiones de modelos significativas.
- La elección del nivel de poda debe adaptarse a la complejidad del conjunto de datos, localizando de manera empírica el punto óptimo de compresión.
- La evaluación debe considerar siempre métricas de FLOPs y *speed-up* además de parámetros, dada su correlación con latencia y consumo energético.

7.5 Trabajo futuro

El trabajo realizado en esta tesis da pie a otros trabajos para continuar en la investigación relacionada a la compresión de modelos. En cuanto al criterio de importancia GradAbs, existen varias posibles extensiones. Una de ellas consiste en introducir ponderaciones multi-clase, en lugar de calcular la importancia solo con respecto a la clase objetivo, podría considerarse un promedio entre las clases más probables o bien un margen entre clases cercanas, lo cual aportaría mayor estabilidad del ranking en situaciones de ambigüedad. Asimismo, sería deseable ponderar la importancia por costo computacional, penalizando la poda de canales que requieran pocos FLOPs y concentrando la reducción en filtros para maximizar el impacto en la eficiencia real.

Otra línea prometedora es la exploración de enfoques híbridos de compresión. Por ejemplo, la combinación de poda con técnicas de cuantificación como PTQ, QAT o esquemas de *mixed-precision*, guiadas por GradAbs. También se plantea la posibilidad de un esquema en dos fases, utilizando distintos métodos de poda, una poda inicial rápida y estable con GradAbs, seguida de una etapa de Regularización creciente para alcanzar reducciones extremas en FLOPs sin comprometer la exactitud.

Por otro lado, en términos de extensión a otros modelos y dominios, se considera explorar la aplicabilidad de los métodos de poda en tareas distintas de la clasificación, tales como la segmentación semántica. De igual modo, adaptar GradAbs a modelos no puramente convolucionales, como Vision Transformers (ViTs), arquitecturas híbridas y redes móviles eficientes.

Finalmente, se reconoce la necesidad de incorporar métricas *hardware-aware* y validación en despliegue real. Los resultados de esta tesis se han basado principalmente en métricas teóricas como FLOPs y factor de aceleración, pero su correlación con el rendimiento práctico no siempre es lineal.

REFERENCIAS

- [1] A. C. Ian Goodfellow, Yoshua Bengio, *Deep Learning*, vol. 521, no. 7553. 2017. doi: 10.1016/B978-0-12-391420-0.09987-X.
- [2] L. Alzubaidi *et al.*, *Review of deep learning: concepts, CNN architectures, challenges, applications, future directions*, vol. 8, no. 1. Springer International Publishing, 2021. doi: 10.1186/s40537-021-00444-8.
- [3] B. L. Deng, G. Li, S. Han, L. Shi, and Y. Xie, "Model Compression and Hardware Acceleration for Neural Networks: A Comprehensive Survey," *Proceedings of the IEEE*, vol. 108, no. 4, pp. 485–532, 2020, doi: 10.1109/JPROC.2020.2976475.
- [4] Z. Li, H. Li, and L. Meng, "Model Compression for Deep Neural Networks: A Survey," *Computers*, vol. 12, no. 3, pp. 1–22, 2023, doi: 10.3390/computers12030060.
- [5] G. Schwalbe and B. Finzel, "A comprehensive taxonomy for explainable artificial intelligence: a systematic survey of surveys on methods and concepts," *Data Min Knowl Discov*, 2023, doi: 10.1007/s10618-022-00867-8.
- [6] S. K. Yeom *et al.*, "Pruning by explaining: A novel criterion for deep neural network pruning," *Pattern Recognit*, vol. 115, p. 107899, 2021, doi: 10.1016/j.patcog.2021.107899.
- [7] M. Sabih, F. Hannig, and J. Teich, "Utilizing Explainable AI for Quantization and Pruning of Deep Neural Networks," 2020. [Online]. Available: <http://arxiv.org/abs/2008.09072>
- [8] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, "Dive into Deep Learning," *Journal of the American College of Radiology*, vol. 17, no. 5, pp. 637–638, Jun. 2021, doi: 10.48550.2106.11342.
- [9] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *32nd International Conference on Machine Learning, ICML 2015*, vol. 1, pp. 448–456, 2015.
- [10] H. Wang, C. Qin, Y. Bai, and Y. Fu, "Why is the State of Neural Network Pruning so Confusing? On the Fairness, Comparison Setup, and Trainability in Network Pruning," 2023.
- [11] C. Xu and J. McAuley, "A Survey on Model Compression and Acceleration for Pretrained Language Models," *Proceedings of the 37th AAAI Conference on Artificial Intelligence, AAAI 2023*, vol. 37, pp. 10566–10575, 2023, doi: 10.1609/aaai.v37i9.26255.
- [12] R. Mishra, H. P. Gupta, and T. Dutta, "A Survey on Deep Neural Network Compression: Challenges, Overview, and Solutions," pp. 1–19, 2020.
- [13] T. Choudhary, V. Mishra, A. Goswami, and J. Sarangapani, *A comprehensive survey on model compression and acceleration*, vol. 53, no. 7. Springer Netherlands, 2020. doi: 10.1007/s10462-020-09816-7.
- [14] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A Survey of Quantization Methods for Efficient Neural Network Inference," *Low-Power Computer Vision*, pp. 291–326, 2022, doi: 10.1201/9781003162810-13.
- [15] B. Rokh, A. Azarpeyvand, and A. Khandteymoori, "A Comprehensive Survey on Model Quantization for Deep Neural Networks in Image Classification," *ACM Trans Intell Syst Technol*, vol. 14, no. 6, pp. 1–48, 2023, doi: 10.1145/3623402.
- [16] H. Wu, P. Judd, X. Zhang, M. Isaev, and P. Micikevicius, "Integer Quantization for Deep Learning Inference: Principles and Empirical Evaluation," pp. 1–20, 2020.
- [17] T. Liang, J. Glossner, L. Wang, S. Shi, and X. Zhang, "Pruning and quantization for deep neural network acceleration: A survey," *Neurocomputing*, vol. 461, pp. 370–403, 2021, doi: 10.1016/j.neucom.2021.07.045.

- [18] N. Ranjan and A. Savakis, "LRP-QViT: Mixed-Precision Vision Transformer Quantization via Layer-wise Relevance Propagation," 2024, [Online]. Available: <http://arxiv.org/abs/2401.11243>
- [19] W. Saeed and C. Omlin, "Explainable AI (XAI): A systematic meta-survey of current challenges and future opportunities," *Knowl Based Syst*, vol. 263, p. 110273, 2023, doi: 10.1016/j.knosys.2023.110273.
- [20] A. Barredo Arrieta *et al.*, "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI," *Information Fusion*, vol. 58, pp. 82–115, 2020, doi: 10.1016/j.inffus.2019.12.012.
- [21] G. Ras, N. Xie, M. van Gerven, and D. Doran, "Explainable Deep Learning: A Field Guide for the Uninitiated," *Journal of Artificial Intelligence Research*, vol. 73, pp. 329–396, 2022, doi: 10.1613/JAIR.1.13200.
- [22] A. Das and P. Rad, "Opportunities and Challenges in Explainable Artificial Intelligence (XAI): A Survey," pp. 1–24, 2020.
- [23] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal Brain Damage (Pruning)," *Adv Neural Inf Process Syst*, pp. 598–605, 1990.
- [24] B. Hassibi and D. Stork, "Second order derivatives for network pruning: Optimal brain surgeon," *Advances in NIPS5*, pp. 164–171, 1993.
- [25] Y. He and L. Xiao, "Structured Pruning for Deep Convolutional Neural Networks: A Survey," *IEEE Trans Pattern Anal Mach Intell*, no. November, 2023, doi: 10.1109/TPAMI.2023.3334614.
- [26] Y. He, P. Liu, Z. Wang, Z. Hu, and Y. Yang, "Filter pruning via geometric median for deep convolutional neural networks acceleration," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, pp. 4335–4344, 2019, doi: 10.1109/CVPR.2019.00447.
- [27] E. Yvinec, A. Dapogny, M. Cord, and K. Bailly, "RED: Looking for Redundancies for Data-Free Structured Compression of Deep Neural Networks," *Adv Neural Inf Process Syst*, vol. 25, no. NeurIPS, pp. 20863–20873, 2021.
- [28] D. Ghimire, D. Kil, and S. H. Kim, "A Survey on Efficient Convolutional Neural Networks and Hardware Acceleration," *Electronics (Switzerland)*, vol. 11, no. 6, 2022, doi: 10.3390/electronics11060945.
- [29] G. C. Marinó, A. Petrini, D. Malchiodi, and M. Frasca, "Deep neural networks compression: A comparative survey and choice recommendations," *Neurocomputing*, vol. 520, pp. 152–170, 2023, doi: 10.1016/j.neucom.2022.11.072.
- [30] K. Soroush, M. Raji, and B. Ghavami, "Compressing Deep Neural Networks Using Explainable AI," *2023 13th International Conference on Computer and Knowledge Engineering, ICCKE 2023*, no. Iccke, pp. 636–641, 2023, doi: 10.1109/ICCKE60553.2023.10326237.
- [31] A. Alqahtani, X. Xie, M. W. Jones, and E. Essa, "Pruning CNN filters via quantifying the importance of deep visual representations," *Computer Vision and Image Understanding*, vol. 208–209, pp. 1–35, 2021, doi: 10.1016/j.cviu.2021.103220.
- [32] K. Yao, F. Cao, Y. Leung, and J. Liang, "Deep neural network compression through interpretability-based filter pruning," *Pattern Recognit*, vol. 119, p. 108056, 2021, doi: 10.1016/j.patcog.2021.108056.
- [33] A. Ganjdanesh, S. Gao, and H. Huang, "Interpretations Steered Network Pruning via Amortized Inferred Saliency Maps," *Lecture Notes in Computer Science (including subseries Lecture Notes in*

- Artificial Intelligence and Lecture Notes in Bioinformatics*), vol. 13681 LNCS, pp. 278–296, 2022, doi: 10.1007/978-3-031-19803-8_17.
- [34] S. Luan, Z. Gu, R. Xu, Q. Zhao, and G. Chen, “LRP-based network pruning and policy distillation of robust and non-robust DRL agents for embedded systems,” *Concurr Comput*, vol. 35, no. 19, pp. 1–14, 2023, doi: 10.1002/cpe.7351.
 - [35] N. Ranjan and A. Savakis, “LRP-QViT: Mixed-Precision Vision Transformer Quantization via Layer-wise Relevance Propagation,” 2024.
 - [36] D. Becking, M. Dreyer, W. Samek, K. Müller, and S. Lapuschkin, “ECQx : Explainability-Driven Quantization for Low-Bit and Sparse DNNs,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 13200 LNAI, pp. 271–296, 2022, doi: 10.1007/978-3-031-04083-2_14.
 - [37] G. Menghani, “Efficient Deep Learning: A Survey on Making Deep Learning Models Smaller, Faster, and Better,” *ACM Comput Surv*, vol. 55, no. 12, 2023, doi: 10.1145/3578938.
 - [38] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pp. 1–14, 2015.
 - [39] O. Russakovsky *et al.*, “ImageNet Large Scale Visual Recognition Challenge,” *Int J Comput Vis*, vol. 115, no. 3, pp. 211–252, 2015, doi: 10.1007/s11263-015-0816-y.
 - [40] M. Simon, E. Rodner, and J. Denzler, “ImageNet pre-trained models with batch normalization,” 2016.
 - [41] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-Decem, pp. 770–778, 2016, doi: 10.1109/CVPR.2016.90.
 - [42] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 2261–2269, 2017, doi: 10.1109/CVPR.2017.243.
 - [43] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A Large-Scale Hierarchical Image Database,” in *IEEE conference on computer vision and pattern recognition*, 2009, pp. 248–255.
 - [44] A. Krizhevsky and G. Hinton, “Learning multiple layers of features from tiny images,” *Toronto, ON, Canada*, Apr. 2009.
 - [45] A. Krizhevsky, “CIFAR-10 and CIFAR-100 datasets,” Department of Computer Science, University of Toronto. [Online]. Available: <https://www.cs.toronto.edu/~kriz/cifar.html>
 - [46] A. Paszke *et al.*, “PyTorch: An imperative style, high-performance deep learning library. In Advances in Neural Information Processing Systems,” *NeurIPS*, no. NeurIPS, pp. 8026–8037, 2019.
 - [47] H. Phan, “PyTorch CIFAR10,” GitHub. [Online]. Available: https://github.com/huyvnphan/PyTorch_CIFAR10
 - [48] G. Fang, X. Ma, M. Song, M. Bi Mi, and X. Wang, “DepGraph: Towards Any Structural Pruning,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2023-June, pp. 16091–16101, 2023, doi: 10.1109/CVPR52729.2023.01544.
 - [49] H. Li *et al.*, “PruningBench: A Comprehensive Benchmark of Structural Pruning,” no. NeurIPS, pp. 1–30, 2024.
 - [50] Y. He and L. Xiao, “Structured Pruning for Deep Convolutional Neural Networks: A Survey,” *IEEE Trans Pattern Anal Mach Intell*, no. November, 2023, doi: 10.1109/TPAMI.2023.3334614.

- [51] H. Li, H. Samet, A. Kadav, I. Durdanovic, and H. P. Graf, "Pruning filters for efficient convnets," *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*, no. 2016, pp. 1–13, 2017.
- [52] Y. He, P. Liu, Z. Wang, Z. Hu, and Y. Yang, "Filter pruning via geometric median for deep convolutional neural networks acceleration," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, pp. 4335–4344, 2019, doi: 10.1109/CVPR.2019.00447.
- [53] M. Lin *et al.*, "Hrank: Filter pruning using high-Rank feature map," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1526–1535, 2020, doi: 10.1109/CVPR42600.2020.00160.
- [54] P. Molchanov, A. Mallya, S. Tyree, I. Frosio, and J. Kautz, "Importance estimation for neural network pruning," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, no. 11264, pp. 11256–11264, 2019, doi: 10.1109/CVPR.2019.01152.
- [55] H. Wang, C. Qin, Y. Zhang, and Y. Fu, "Neural Pruning Via Growing Regularization," *ICLR 2021 - 9th International Conference on Learning Representations*, no. 1990, pp. 1–16, 2021.
- [56] R. R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, and D. Batra, "Grad-CAM: Why did you say that?," pp. 1–4, 2016, [Online]. Available: <http://arxiv.org/abs/1611.07450>
- [57] A. Chattopadhyay, A. Sarkar, P. Howlader, and V. N. Balasubramanian, "Grad-CAM++: Generalized gradient-based visual explanations for deep convolutional networks," *Proceedings - 2018 IEEE Winter Conference on Applications of Computer Vision, WACV 2018*, vol. 2018-Janua, pp. 839–847, 2018, doi: 10.1109/WACV.2018.00097.
- [58] H. Wang *et al.*, "Score-CAM: Score-Weighted Visual Explanations for Convolutional Neural Networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2020, pp. 24–25.
- [59] G. Localization and S. Desai, "Ablation-CAM : Visual Explanations for Deep Convolutional Network via," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2020, pp. 983–991.
- [60] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2921–2929, 2016.
- [61] R. R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, and D. Batra, "Grad-CAM: Why did you say that?," pp. 1–4, 2016.
- [62] R. L. Draelos and L. Carin, "Use HiResCAM instead of Grad-CAM for faithful explanations of convolutional neural networks," pp. 1–20, 2020, [Online]. Available: <http://arxiv.org/abs/2011.08891>
- [63] A. Chattopadhyay, A. Sarkar, P. Howlader, and V. N. Balasubramanian, "Grad-CAM++: Generalized gradient-based visual explanations for deep convolutional networks," *Proceedings - 2018 IEEE Winter Conference on Applications of Computer Vision, WACV 2018*, vol. 2018-Janua, pp. 839–847, 2018, doi: 10.1109/WACV.2018.00097.
- [64] R. Fu, Q. Hu, X. Dong, Y. Guo, Y. Gao, and B. Li, "Axiom-based Grad-CAM: Towards Accurate Visualization and Explanation of CNNs," *31st British Machine Vision Conference, BMVC 2020*, 2020.

- [65] G. Localization and S. Desai, "Ablation-CAM : Visual Explanations for Deep Convolutional Network via," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2020, pp. 983–991.
- [66] H. Wang *et al.*, "Score-CAM: Score-Weighted Visual Explanations for Convolutional Neural Networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2020, pp. 24–25.
- [67] M. Bany Muhammad and M. Yeasin, "Eigen-CAM: Visual Explanations for Deep Convolutional Neural Networks," *SN Comput Sci*, vol. 2, no. 1, Feb. 2021, doi: 10.1007/s42979-021-00449-3.
- [68] P. T. Jiang, C. Bin Zhang, Q. Hou, M. M. Cheng, and Y. Wei, "LayerCAM: Exploring hierarchical class activation maps for localization," *IEEE Transactions on Image Processing*, vol. 30, no. 8, pp. 5875–5888, 2021, doi: 10.1109/TIP.2021.3089943.
- [69] S. Srinivas and F. Fleuret, "Full-gradient representation for neural network visualization," *Adv Neural Inf Process Syst*, vol. 32, no. NeurIPS, pp. 1–10, 2019.
- [70] S. Karmani, T. Sivakaran, G. Prasad, M. Ali, W. Yang, and S. Tang, "KPCA-CAM: Visual Explainability of Deep Computer Vision Models Using Kernel PCA," *2024 IEEE 26th International Workshop on Multimedia Signal Processing, MMSP 2024*, 2024, doi: 10.1109/MMSP61759.2024.10743968.
- [71] K. Ahmed Asif Fuad, P. E. Martin, R. Giot, R. Bourqui, J. Benois-Pineau, and A. Zemhari, "Features Understanding in 3D CNNs for Actions Recognition in Video," *2020 10th International Conference on Image Processing Theory, Tools and Applications, IPTA 2020*, 2020, doi: 10.1109/IPTA50016.2020.9286629.
- [72] Z. Zhang *et al.*, "Finer-CAM: Spotting the Difference Reveals Finer Details for Visual Explanation," 2025.
- [73] D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba, "Network dissection: Quantifying interpretability of deep visual representations," *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-January, pp. 3319–3327, 2017, doi: 10.1109/CVPR.2017.354.
- [74] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," *Adv Neural Inf Process Syst*, vol. 4, no. January, pp. 3320–3328, 2014.
- [75] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8689 LNCS, no. PART 1, pp. 818–833, 2014, doi: 10.1007/978-3-319-10590-1_53.
- [76] W. Samek, A. Binder, G. Montavon, S. Lapuschkin, and K. R. Müller, "Evaluating the visualization of what a deep neural network has learned," *IEEE Trans Neural Netw Learn Syst*, vol. 28, no. 11, pp. 2660–2673, 2017, doi: 10.1109/TNNLS.2016.2599820.
- [77] G. Montavon, W. Samek, and K. Müller, "Methods for interpreting and understanding deep neural networks," *Digit Signal Process*, vol. 73, pp. 1–15, 2018, doi: 10.1016/j.dsp.2017.10.011.
- [78] A. Arias-Duart, F. Parés, D. Garcia-Gasulla, and V. Gimenez-Abalos, "Focus! Rating XAI Methods and Finding Biases," in *2022 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, Padua, Italy, Feb. 2022, pp. 1–8. doi: 10.1109/FUZZ-IEEE55066.2022.9882821.
- [79] M. Bello, R. Amador, M. M. García, J. Del Ser, P. Mesejo, and Ó. Cordón, "The level of strength of an explanation: A quantitative evaluation technique for post-hoc XAI methods," *Pattern Recognit*, vol. 161, no. November 2024, p. 111221, 2025, doi: 10.1016/j.patcog.2024.111221.

- [80] G. Fang, X. Ma, M. Song, M. Bi Mi, and X. Wang, “DepGraph: Towards Any Structural Pruning,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2023-June, pp. 16091–16101, 2023, doi: 10.1109/CVPR52729.2023.01544.
- [81] N. Lee, T. Ajanthan, and P. H. S. Torr, “SnIP: Single-shot network pruning based on connection sensitivity,” *7th International Conference on Learning Representations, ICLR 2019*, 2019.
- [82] C. Wang, G. Zhang, and R. Grosse, “Picking Winning Tickets Before Training By Preserving Gradient Flow,” *8th International Conference on Learning Representations, ICLR 2020*, no. 2019, pp. 1–11, 2020.
- [83] H. Tanaka, D. Kunin, D. L. K. Yamins, and S. Ganguli, “Pruning neural networks without any data by iteratively conserving synaptic flow,” *Adv Neural Inf Process Syst*, vol. 2020-Decem, no. NeurIPS, 2020.

ANEXOS

A Resultados de selectividad

Tabla A.1. Resultados de la métrica de selectividad para cada método CAM evaluado en 10 imágenes (Etapa 1 de ResNet-50). Valores menores son mejores.

	Imagen 1	Imagen 2	Imagen 3	Imagen 4	Imagen 5	Imagen 6	Imagen 7	Imagen 8	Imagen 9	Imagen 10	Media ↓	Std↓	Rank ↓
AblationCAM	5.0978	226.0349	6.1091	202.4963	82.1218	21.1888	135.2953	89.6642	128.0474	430.6897	132.6745	123.1104	7
EigenCAM	3.5903	330.3882	140.92	118.3611	161.0093	53.2759	135.1975	78.3498	213.0169	117.5991	135.1708	85.27056	8
EigenGradCAM	4.3561	88.5577	2.2761	166.9651	75.0824	30.3627	32.8987	24.8748	22.7505	199.3171	64.74412	65.06242	2
FEM	4.4345	260.9065	39.2717	189.1103	243.4703	66.0515	148.0393	190.1532	65.4133	117.8973	132.4748	83.54297	6
FinerCAM	6.547	82.4918	4.1247	181.6278	207.7768	35.5782	102.2562	205.7187	187.4893	479.1801	149.2791	133.7873	10
FullGrad	4.7945	157.5677	4.0553	316.5847	123.244	16.5107	187.5439	41.7037	20.3515	197.6329	106.9989	101.2185	5
GradCAM	6.5675	82.6863	4.1247	181.5831	207.7561	35.5749	102.2562	205.7213	187.6657	479.1971	149.3133	133.783	11
GradCAM++	2.9996	328.6716	76.5317	172.2103	253.9898	134.7563	136.0264	155.4722	85.1243	171.0588	151.6841	86.7865	12
HiResCAM	1.2047	40.8344	2.654	149.1079	109.8774	12.1481	32.6428	13.2889	33.7441	207.9841	60.34864	67.34118	1
KPCA_CAM	3.5727	262.844	140.9821	118.5753	491.7436	147.5779	136.7594	325.9854	43.638	117.376	178.9054	136.8233	13
LayerCAM	4.0029	103.4894	3.0221	262.2283	114.782	19.7639	105.8331	20.8956	18.644	124.7704	77.74317	77.57379	3
ScoreCAM	1.7074	193.382	5.0498	151.4532	101.8377	18.2715	149.2023	88.5768	27.7318	316.7098	105.3922	95.48505	4
XGradCAM	5.4006	165.7261	5.4299	134.5993	225.4831	34.9276	106.6159	197.2167	151.4243	418.0181	144.4842	116.9942	9

Tabla A.2. Resultados de la métrica de selectividad para cada método CAM evaluado en 10 imágenes (Etapa 2 de ResNet-50). Valores menores son mejores.

	Imagen 1	Imagen 2	Imagen 3	Imagen 4	Imagen 5	Imagen 6	Imagen 7	Imagen 8	Imagen 9	Imagen 10	Media ↓	Std↓	Rank ↓
AblationCAM	1.2312	228.7336	4.1095	127.2645	76.1327	12.5998	147.4814	155.4421	80.0091	413.7771	124.6781	119.402	7
EigenCAM	5.2569	96.7083	12.7616	113.2573	123.5724	143.3687	99.8255	343.037	75.7236	426.1701	143.9681	128.8104	12
EigenGradCAM	1.7304	162.578	1.7706	147.8543	104.0358	21.8974	164.7909	176.5013	24.0679	196.2906	100.1517	75.30997	5
FEM	2.2615	55.6528	9.5031	173.6202	84.1566	60.3649	164.0541	26.2293	64.1182	118.5482	75.85089	56.82944	4
FinerCAM	1.9304	156.1942	8.9007	165.6857	83.5695	22.0149	136.9082	157.0422	28.6126	496.175	125.7033	138.4922	11
FullGrad	4.7945	157.5676	4.0552	316.591	123.2383	16.5107	187.5437	41.7037	20.3515	197.6304	106.9987	101.2195	6
GradCAM	1.9303	156.1938	8.9015	165.643	83.5695	22.0149	136.9082	157.0544	28.6125	496.1664	125.6995	138.4889	10
GradCAM++	2.3581	50.5923	7.2237	179.9159	83.3251	58.1215	170.4573	24.1828	57.0997	113.2728	74.65492	59.41908	3
HiResCAM	1.369	59.6544	6.3569	175.0125	81.6665	14.5338	46.9919	23.4112	15.1966	88.7329	51.29257	50.65813	1
KPCA_CAM	4.4411	98.0578	12.5935	113.3566	338.7159	78.2852	190.291	343.2723	219.0941	88.7246	148.6832	115.0453	13
LayerCAM	1.3169	94.3021	2.6988	186.1566	127.0324	17.9943	72.276	26.8641	21.6206	90.949	64.12108	57.95408	2
ScoreCAM	1.4584	93.4803	2.9715	290.1987	132.5909	30.2829	106.0987	153.4299	45.0069	401.0077	125.6526	122.9662	9
XGradCAM	1.6613	87.2882	6.2186	128.8956	77.1849	18.4406	102.5635	288.794	34.6342	504.9974	125.0678	149.7651	8

Tabla A.3. Resultados de la métrica de selectividad para cada método CAM evaluado en 10 imágenes (Etapa 3 de ResNet-50). Valores menores son mejores.

	Imagen 1	Imagen 2	Imagen 3	Imagen 4	Imagen 5	Imagen 6	Imagen 7	Imagen 8	Imagen 9	Imagen 10	Media ↓	Std↓	Rank ↓
AblationCAM	1.8926	373.0994	1.7758	133.3339	46.8014	17.158	101.5763	41.6398	54.9921	85.7312	85.8001	104.0585	5
EigenCAM	3.8706	274.3988	99.0237	213.2001	184.8105	76.5538	183.7536	18.8947	37.0443	142.2644	123.381	86.07501	11
EigenGradCAM	3.1868	183.969	3.6372	187.9385	200.0558	30.2203	322.9627	21.7021	24.5184	573.8633	155.205	174.3369	13
FEM	7.0283	83.9368	8.1821	327.8944	72.0783	29.5988	145.0381	40.3413	54.032	142.502	91.0632	91.49454	7
FinerCAM	1.8039	345.8999	1.9242	185.2526	64.8829	24.3094	124.6292	61.3326	32.1162	181.5462	102.37	103.5674	9
FullGrad	4.7945	157.5677	4.0553	316.6125	123.2383	16.5107	187.5437	41.7037	20.3515	197.6182	107	101.2228	10
GradCAM	1.8039	345.8999	1.9242	185.2526	64.8829	24.3094	124.6292	61.3325	32.1162	181.5462	102.37	103.5674	8
GradCAM++	6.4256	76.357	5.5179	268.6614	54.1386	23.0579	176.1077	32.7496	51.1435	139.809	83.3968	81.38925	4
HiResCAM	3.987	147.7358	1.6933	150.3377	63.2659	12.0854	148.1435	25.4313	30.4775	70.6566	65.3814	58.61902	1
KPCA_CAM	3.8612	274.609	53.3633	212.9231	184.862	78.2396	183.2967	341.1618	37.0695	141.6142	151.1	103.4456	12
LayerCAM	3.0232	107.2907	2.5069	173.3916	114.6554	17.522	127.137	25.52	18.8708	116.8604	70.6778	59.93186	2
ScoreCAM	2.3381	88.6955	2.2744	266.5662	97.2945	20.8423	174.8367	32.0414	37.941	169.1202	89.195	84.21036	6
XGradCAM	1.7593	307.3009	2.172	125.8512	80.2274	41.7686	77.8131	34.5856	56.4013	92.9311	82.0811	83.75953	3

Tabla A.4. Resultados de la métrica de selectividad para cada método CAM evaluado en 10 imágenes (Etapa 4 de ResNet-50). Valores menores son mejores.

	Imagen 1	Imagen 2	Imagen 3	Imagen 4	Imagen 5	Imagen 6	Imagen 7	Imagen 8	Imagen 9	Imagen 10	Media ↓	Std↓	Rank ↓
AblationCAM	2.792	137.7735	3.9602	400.4562	166.9	71.7992	217.8315	37.1173	37.1311	241.4674	131.7228	121.4024	7
EigenCAM	5.8763	289.0829	3.1977	466.5695	127.9462	84.408	523.6558	49.1881	28.7352	549.0799	212.774	212.3717	13
EigenGradCAM	2.7973	191.8677	4.8665	360.2933	146.5986	66.2566	527.0289	48.2364	47.6408	550.9427	194.6529	199.8833	12
FEM	6.6141	184.3281	2.5629	438.2459	180.5953	80.1836	255.768	39.7036	57.538	316.6186	156.2158	138.4372	10
FinerCAM	3.1036	129.9107	3.1139	405.3668	149.117	46.7865	209.5306	27.9132	35.6212	303.3337	131.3797	130.7667	3
FullGrad	4.7945	157.5676	4.0553	316.5909	123.244	16.5107	187.5437	41.7037	20.3511	197.6223	106.9984	101.2188	1
GradCAM	3.1036	129.9107	3.1139	405.3668	149.117	46.7865	209.5306	27.9132	35.6212	303.3337	131.3797	130.7667	3
GradCAM++	4.0178	128.1055	3.3043	392.6827	158.9949	58.1333	224.8013	31.5109	31.8659	279.4281	131.2845	125.5247	2
HiResCAM	3.1036	129.9107	3.1139	405.3668	149.117	46.7865	209.5306	27.9132	35.6212	303.3337	131.3797	130.7667	3
KPCA_CAM	5.776	240.3048	3.2563	470.4831	127.7327	85.7553	290.4503	45.5472	29.8158	394.9728	169.4094	160.9805	11
LayerCAM	4.3144	131.9589	3.2695	395.3574	158.3143	66.8054	227.4349	31.6701	31.6084	277.116	132.7849	125.491	8
ScoreCAM	2.5513	166.198	3.0117	497.5268	173.365	51.901	234.9212	57.7696	49.1533	266.4269	150.2825	146.286	9
XGradCAM	3.1036	129.9107	3.1139	405.3668	149.117	46.7865	209.5306	27.9132	35.6212	303.3337	131.3797	130.7667	3

B Resultados de continuidad

Tabla B.1. Resultados de la métrica de continuidad para cada método CAM evaluado en 10 imágenes (Etapa 1 de ResNet-50). Valores menores son mejores.

	Imagen 1	Imagen 2	Imagen 3	Imagen 4	Imagen 5	Imagen 6	Imagen 7	Imagen 8	Imagen 9	Imagen 10	Media ↓	Std↓	Rank ↓
AblationCAM	21.0482	20.5434	27.6614	10.349	41.1328	52.8559	9.6638	13.8234	30.9672	37.4748	26.552	13.4933	11
EigenCAM	33.3456	24.9527	55.3886	20.9821	52.0116	39.0414	22.7985	23.3978	21.2898	42.8307	33.6039	12.3891	13
EigenGradCAM	4.3527	0.6852	1.7179	2.6018	2.7163	2.6953	1.7938	0.9788	2.3367	4.4412	2.43197	1.18298	1
FEM	14.3022	15.8065	17.0599	15.9895	20.7813	20.2289	10.4661	12.1077	17.7863	19.1719	16.37	3.2007	8
FinerCAM	18.8965	13.8595	14.1473	14.268	19.8737	5.1607	12.3435	15.7824	5.0637	13.2199	13.2615	4.66536	6
FullGrad	23.8511	20.9893	23.669	26.9018	30.7261	28.0813	17.0777	16.3209	24.5518	29.3928	24.1562	4.64267	10
GradCAM	18.8964	13.8594	14.1483	14.2681	19.8695	5.1606	12.3431	15.7829	5.0639	13.2225	13.2615	4.66479	5
GradCAM++	15.6581	7.6363	15.7465	11.7907	12.8963	23.2824	8.8627	12.1119	13.5037	16.6811	13.817	4.19064	7
HiResCAM	6.0692	1.9934	2.7054	3.718	4.0298	4.8117	2.1516	1.7921	3.4655	5.008	3.57447	1.36036	2
KPCA_CAM	31.3875	20.8597	36.9253	20.9681	32.6369	54.2336	17.2211	19.9793	30.1882	30.8878	29.5288	10.3515	12
LayerCAM	8.4368	3.6854	4.9913	5.9987	8.7178	7.7628	4.1139	3.9407	6.4357	7.0273	6.11104	1.7812	3
ScoreCAM	21.7294	15.6023	23.9367	11.6093	17.8451	42.8144	16.9526	14.9748	23.1942	24.4045	21.3063	8.24894	9
XGradCAM	18.1416	10.848	4.8381	11.2727	12.9541	24.742	9.7843	3.2046	20.2046	4.8153	12.0805	6.74741	4

Tabla B.2. Resultados de la métrica de continuidad para cada método CAM evaluado en 10 imágenes (Etapa 2 de ResNet-50). Valores menores son mejores.

	Imagen 1	Imagen 2	Imagen 3	Imagen 4	Imagen 5	Imagen 6	Imagen 7	Imagen 8	Imagen 9	Imagen 10	Media ↓	Std↓	Rank ↓
AblationCAM	21.4501	10.9114	30.9349	12.5908	24.3806	24.8149	12.9516	12.7216	18.2448	28.1541	19.7155	6.8772	10
EigenCAM	33.6363	23.5739	44.8247	32.7464	40.9408	39.2375	19.0197	26.4853	36.8259	32.9414	33.0232	7.6396	13
EigenGradCAM	12.0427	7.6483	12.6883	4.6808	8.0823	11.81	5.1117	8.0176	10.3016	19.3838	9.97671	4.1005	2
FEM	17.7217	14.4237	22.1818	16.4632	19.2365	19.6692	10.0442	16.1202	20.6049	23.6888	18.0154	3.7816	8
FinerCAM	22.6509	10.4255	6.2008	15.4139	17.5662	17.7702	12.0669	10.5359	18.573	10.6621	14.1865	4.7415	6
FullGrad	23.8511	20.9893	23.669	26.9018	30.7262	28.0813	17.0776	16.3209	24.5517	29.3927	24.1562	4.6427	12
GradCAM	22.6507	10.4256	6.2008	15.4137	17.5661	17.7701	12.067	10.5357	18.5727	10.662	14.1864	4.7414	5
GradCAM++	16.4756	13.4759	19.0834	16.6341	17.1251	18.7421	8.9932	14.7903	19.9058	22.3737	16.7599	3.549	7
HiResCAM	7.4781	4.7524	4.7563	7.2034	7.217	5.3334	4.0106	4.4376	5.4192	5.1338	5.57418	1.1973	1
KPCA_CAM	34.3957	14.5292	21.8168	17.4483	31.7953	31.0058	13.8977	16.1115	19.4948	27.0704	22.7566	7.3065	11
LayerCAM	13.1345	9.1297	9.5086	12.8201	13.4052	12.4814	7.6497	6.9167	11.0565	11.2968	10.7399	2.211	3
ScoreCAM	27.2931	16.318	15.0789	15.8443	26.1889	21.4015	16.6843	19.7593	16.7821	20.961	19.6311	4.1167	9
XGradCAM	18.922	8.5763	7.4254	11.3067	13.212	8.8883	11.5902	5.5177	14.4137	19.9175	11.977	4.5043	4

Tabla B.3. Resultados de la métrica de continuidad para cada método CAM evaluado en 10 imágenes (Etapa 3 de ResNet-50). Valores menores son mejores.

	Imagen 1	Imagen 2	Imagen 3	Imagen 4	Imagen 5	Imagen 6	Imagen 7	Imagen 8	Imagen 9	Imagen 10	Media↓	Std↓	Rank↓
AblationCAM	19.2423	9.7268	17.0244	13.7294	27.3759	21.2305	10.7117	13.7539	21.263	25.0812	17.9139	5.6514	9
EigenCAM	19.2597	21.9565	24.9744	20.9121	37.0657	60.9998	15.2369	18.079	39.108	28.7133	28.6305	13.133	13
EigenGradCAM	4.932	3.5769	5.187	7.5736	5.9664	5.0686	5.0966	4.193	27.7688	3.8763	7.32392	6.899	1
FEM	19.4459	17.9745	18.4972	13.7404	22.937	21.8711	10.333	14.0846	15.7318	25.0141	17.963	4.3392	10
FinerCAM	20.2746	11.0299	15.2701	12.4054	16.5626	15.791	9.7069	12.1556	15.9064	26.1894	15.5292	4.6129	6
FullGrad	23.851	20.9893	23.669	26.9019	30.7256	28.0813	17.0777	16.3208	24.5518	29.3927	24.1561	4.6426	12
GradCAM	20.2746	11.0299	15.2701	12.4054	16.5621	15.791	9.7069	12.1556	15.9064	26.1894	15.5291	4.6129	5
GradCAM++	23.4507	17.0775	18.6122	13.5496	19.2316	20.088	8.9511	12.7267	16.5459	26.4317	17.6665	4.8827	8
HiResCAM	10.7675	4.885	8.3144	9.3582	11.0504	6.17	4.8288	5.3611	8.0254	12.4805	8.12413	2.6249	2
KPCA_CAM	19.2906	12.1434	8.0231	12.7288	9.8234	18.7592	15.4042	15.8435	16.8155	28.4276	15.7259	5.4821	7
LayerCAM	15.9598	11.3889	10.6662	17.9042	17.245	11.6228	8.8779	11.0602	14.4362	21.5919	14.0753	3.819	4
ScoreCAM	21.8635	11.7115	30.1189	14.6843	21.8879	11.2325	12.4591	16.9005	22.969	27.2424	19.107	6.3484	11
XGradCAM	13.1102	6.7992	15.1539	11.6123	16.0082	13.1745	7.1798	8.4706	13.7384	18.0088	12.3256	3.5997	3

C Resultados de enfoque

Tabla C.1. Resultados de la métrica de enfoque para cada método CAM evaluado en 10 imágenes (Etapa 4 de ResNet-50). Valores mayores son mejores.

	Mosaico 1	Mosaico 2	Mosaico 3	Mosaico 4	Mosaico 5	Mosaico 6	Mosaico 7	Mosaico 8	Mosaico 9	Mosaico 10	Media↑	Std↓	Rank↓
AblationCAM	0.909	0.731	0.711	0.832	0.789	0.943	0.834	0.976	0.74	0.924	0.8389	0.0906	6
EigenCAM	0.743	0.124	0.87	0.946	0.898	0.921	0.074	0.144	0.007	1	0.5727	0.4025	11
EigenGradCAM	0.968	0.137	0.901	0.949	0.933	0.99	0.998	1	0.989	1	0.8865	0.2518	1
FEM	0.366	0.391	0.54	0.494	0.563	0.413	0.47	0.488	0.498	0.574	0.4797	0.0674	12
FinerCAM	0.952	0.77	0.769	0.874	0.813	0.967	0.905	0.993	0.784	0.968	0.8795	0.0849	2
FullGrad	0.574	0.593	0.613	0.722	0.635	0.713	0.713	0.704	0.625	0.664	0.6556	0.0521	10
GradCAM	0.952	0.77	0.769	0.874	0.813	0.967	0.905	0.993	0.784	0.968	0.8795	0.0849	2
GradCAM++	0.705	0.591	0.684	0.806	0.762	0.849	0.821	0.823	0.631	0.832	0.7504	0.0872	8
HiResCAM	0.952	0.77	0.769	0.874	0.813	0.967	0.905	0.993	0.784	0.968	0.8795	0.0849	2
KPCA_CAM	0.094	0.162	0.834	0.937	0.895	0.074	0.082	0.136	0.234	1	0.4448	0.3894	13
LayerCAM	0.648	0.573	0.676	0.795	0.749	0.82	0.797	0.788	0.627	0.793	0.7266	0.0832	9
ScoreCAM	0.887	0.754	0.735	0.843	0.827	0.904	0.826	0.901	0.759	0.859	0.8295	0.0589	7
XGradCAM	0.952	0.77	0.769	0.874	0.813	0.967	0.905	0.993	0.784	0.968	0.8795	0.0849	2

D Resultados de LSE

Tabla D.1. Resultados de la métrica de LSE para cada método CAM evaluado en 10 imágenes (Etapa 1 de ResNet-50). Valores mayores son mejores.

	Imagen 1	Imagen 2	Imagen 3	Imagen 4	Imagen 5	Imagen 6	Imagen 7	Imagen 8	Imagen 9	Imagen 10	Media↑	Std↓	Rank↓
AblationCAM	2.5145	2.6754	2.2469	2.4243	2.8153	3.1403	2.1354	3.0391	0	2.4497	2.3441	0.8392	6
EigenCAM	1	1	1	1	1	1	1	1	1	1	1	0	10
EigenGradCAM	4.2164	4.6384	4.8111	3.7313	4.4106	4.1116	3.789	4.6906	5.1792	4.4008	4.3979	0.4307	2
FEM	1	1	1	1	1	1	1	1	1	1	1	0	10
FinerCAM	3.5936	3.8776	4.1842	3.4791	3.5682	3.6473	3.6353	4.1319	3.874	3.8695	3.7861	0.2285	4
FullGrad	1.3565	1.6301	1.0192	1.1987	1.2467	1.1054	1.2125	1.1798	1.3045	1.2565	1.251	0.1556	9
GradCAM	3.5936	3.876	4.1839	3.479	3.5683	3.6471	3.6348	4.1314	3.874	3.8698	3.7858	0.2283	5
GradCAM++	0.9815	0.9542	0.9188	1.0151	0.9254	0.9484	0.9524	0.9798	0.9783	0.9412	0.9595	0.0277	13
HiResCAM	5.0534	5.4412	4.2307	4.5129	5.5703	4.1885	5.3872	4.1631	4.6461	5.1711	4.8365	0.5242	1
KPCA_CAM	1	1	1	1	1	1	1	1	1	1	1	0	10
LayerCAM	2.2353	2.5026	1.487	2.3493	2.1527	1.4544	2.042	1.5282	2.1224	1.5609	1.9435	0.3761	8
ScoreCAM	1.7807	2.1347	2.3015	1.7757	1.7402	1.9303	2.1256	2.327	2.2892	1.6556	2.0061	0.2458	7
XGradCAM	3.6996	3.6664	3.9696	3.6643	3.872	3.0441	4.0286	4.4901	3.8541	3.9251	3.8214	0.3456	3

Tabla D.2. Resultados de la métrica de LSE para cada método CAM evaluado en 10 imágenes (Etapa 2 de ResNet-50). Valores mayores son mejores.

	Imagen 1	Imagen 2	Imagen 3	Imagen 4	Imagen 5	Imagen 6	Imagen 7	Imagen 8	Imagen 9	Imagen 10	Media ↑	Std ↓	Rank ↓
AblationCAM	2.4852	2.9118	2.2913	2.2547	2.7771	2.8529	2.7184	2.5894	2.9103	2.5287	2.632	0.2292	6
EigenCAM	1	1	1	1	1	1	1	1	1	1	1	0	11
EigenGradCAM	2.497	4.9037	1.4146	5.354	3.0778	1.465	2.4459	1.8538	3.3531	13.9261	4.0291	3.5331	1
FEM	1	1	1	1	1	1	1	1	1	1	1	0	11
FinerCAM	3.4899	4.5838	3.3876	4.0742	3.8008	2.8244	3.1714	3.417	3.4645	4.262	3.6476	0.5043	3
FullGrad	1.3565	1.6301	1.0192	1.1987	1.2466	1.1054	1.2125	1.1798	1.3046	1.2565	1.251	0.1556	8
GradCAM	3.4899	4.5838	3.3876	4.0742	3.8008	2.8244	3.1714	3.417	3.4645	4.2621	3.6476	0.5043	2
GradCAM++	1.0059	1.0287	1.0779	0.9865	1.0262	0.9413	0.9677	1.0278	1.0376	1.0774	1.0177	0.0416	10
HiResCAM	3.8624	2.6873	3.1366	5.1812	3.8369	2.935	3.6193	3.4138	4.1893	3.4221	3.6284	0.6724	5
KPCA_CAM	1	1	1	1	1	1	1	1	1	1	1	0	11
LayerCAM	1.3936	1.1245	1.1206	1.5102	1.02	0.9713	1.2287	0.9137	1.4306	1.1717	1.1885	0.1913	9
ScoreCAM	1.7474	2.1417	2.0589	1.8941	1.8993	1.1351	1.8418	1.7452	1.9795	1.9508	1.8394	0.263	7
XGradCAM	4.538	2.8897	3.557	4.1696	3.7631	3.0923	3.3566	3.4531	3.7925	3.7368	3.6349	0.4604	4

Tabla D.3. Resultados de la métrica de LSE para cada método CAM evaluado en 10 imágenes (Etapa 3 de ResNet-50). Valores mayores son mejores.

	Imagen 1	Imagen 2	Imagen 3	Imagen 4	Imagen 5	Imagen 6	Imagen 7	Imagen 8	Imagen 9	Imagen 10	Media ↑	Std ↓	Rank ↓
AblationCAM	2.2149	2.4952	2.5845	2.8151	2.9658	2.2522	2.2326	3.0318	2.7702	2.8516	2.6214	0.2949	6
EigenCAM	1	1	1	1	1	1	1	1	1	1	1	0	10
EigenGradCAM	4.7647	7.0334	4.5609	7.1556	4.0169	12.3324	9.3458	8.2669	4.9051	2.6702	6.5052	2.7586	1
FEM	1	1	1	1	1	1	1	1	1	1	1	0	10
FinerCAM	3.3886	4.024	4.0652	3.8673	3.3581	3.486	3.6451	4.2445	3.8489	4.7772	3.8705	0.4144	4
FullGrad	1.3565	1.6301	1.0192	1.1987	1.2466	1.1054	1.2125	1.1798	1.3046	1.2565	1.251	0.1556	9
GradCAM	3.3887	4.024	4.0652	3.8673	3.3581	3.4861	3.6451	4.2445	3.8489	4.7774	3.8705	0.4144	3
GradCAM++	0.9273	0.9507	0.9938	1.0346	0.9883	0.9601	1.0194	0.9681	1.0118	0.9915	0.9846	0.0315	13
HiResCAM	4.3202	4.8759	2.8255	4.2061	5.0567	2.1475	5.369	4.0885	4.8833	2.7433	4.0516	1.0514	2
KPCA_CAM	1	1	1	1	1	1	1	1	1	1	1	0	10
LayerCAM	1.474	1.7688	1.1814	1.3964	1.2198	0.986	1.3709	1.0244	1.4329	1.1128	1.2967	0.2263	8
ScoreCAM	1.9269	1.8219	1.3068	1.5172	2.1757	1.5959	1.9011	1.9557	2.1257	2.2148	1.8542	0.2839	7
XGradCAM	4.1634	3.8435	3.5311	3.8639	3.9532	2.5169	3.6512	3.9832	3.5187	5.3904	3.8416	0.6728	5

Tabla D.4. Resultados de la métrica de LSE para cada método CAM evaluado en 10 imágenes (Etapa 4 de ResNet-50). Valores mayores son mejores.

	Imagen 1	Imagen 2	Imagen 3	Imagen 4	Imagen 5	Imagen 6	Imagen 7	Imagen 8	Imagen 9	Imagen 10	Media ↑	Std ↓	Rank ↓
AblationCAM	1.3475	1.8389	1.4283	1.4606	1.7166	1.526	1.5576	1.6939	1.4964	1.7844	1.585	0.1555	5
EigenCAM	1	1	1	1	1	1	1	1	1	1	1	0	11
EigenGradCAM	1.986	2.0086	1.0315	1.2314	1.5113	1.0701	1.0461	1.0859	0.9832	1.2073	1.3161	0.3693	6
FEM	1	1	1	1	1	1	1	1	1	1	1	0	11
FinerCAM	3.2761	3.6573	3.2953	3.3752	3.9129	3.0694	3.2672	3.5034	3.4042	3.6436	3.4405	0.2317	1
FullGrad	1.3565	1.6301	1.0192	1.1987	1.2467	1.1054	1.2125	1.1798	1.3046	1.2565	1.251	0.1556	7
GradCAM	3.2761	3.6573	3.2953	3.3752	3.9129	3.0694	3.2672	3.5034	3.4042	3.6436	3.4405	0.2317	1
GradCAM++	1.0652	1.1398	0.9815	1.0607	1.2019	0.9585	1.0466	1.1162	0.9672	1.1245	1.0662	0.0766	9
HiResCAM	3.2761	3.6573	3.2953	3.3752	3.9129	3.0694	3.2672	3.5034	3.4042	3.6436	3.4405	0.2317	1
KPCA_CAM	1	1	1	1	1	1	1	1	1	1	1	0	11
LayerCAM	1.0496	1.0929	0.9694	1.0297	1.1449	0.9572	1.0225	1.0907	0.9458	1.0963	1.0399	0.0638	10
ScoreCAM	1.2503	1.3077	1.0192	1.4724	1.3999	1.3083	1.1427	1.0923	1.2328	1.2706	1.2496	0.13	8
XGradCAM	3.2761	3.6573	3.2953	3.3752	3.9129	3.0694	3.2672	3.5034	3.4042	3.6436	3.4405	0.2317	1

GLOSARIO Y NOTACIÓN MATEMÁTICA

DNNs	Redes Neuronales Profundas (<i>Deep Neural Networks</i>)
XAI	Inteligencia Artificial Explicable (<i>Explainable Artificial Intelligence</i>)
CNNs	Redes Neuronales Convolucionales (<i>Convolutional Neural Networks</i>)
DL	Aprendizaje Profundo (<i>Deep Learning</i>)
ML	Aprendizaje Automático (<i>Machine Learning</i>)
RNNs	Redes Neuronales Recurrentes (<i>Recurrent Neural Networks</i>)
IA	Inteligencia Artificial
$T^{(l)}$	Tensor de entrada de la l-ésima capa
$F_k^{(l)}$	k-ésimo filtro de la l-ésima capa
$K^{(l)}$	Kernel de la l-ésima capa
$L^{(l)}$	l-ésima capa
$M_k^{(l)}$	k-ésimo mapa de activación/características de la l-ésima capa
$m_k^{(l)}(x, y)$	Activación del k-ésimo mapa de características de la l-ésima capa en la posición espacial (x, y)
W	Parámetros de un modelo de DNN
$f(x; \mathbf{W})$	Modelo de DNN
D	Conjunto de datos
$B = \{x_1, \dots, x_m\}$	Mini-lote
$C_q^c(x, y)$	Mapa de activación de clase (CAM) obtenido por el método q para la clase c