Clasificación de Imágenes con Arquitecturas Ligeras de Redes Neuronales en el Espacio de Fourier

Por:

Daniel Lima López

Tesis sometida como requisito parcial
para obtener el grado de:

MAESTRÍA EN CIENCIAS EN LA ESPECIALIDAD DE CIENCIAS COMPUTACIONALES

en el

Instituto Nacional de Astrofísica, Óptica y Electrónica

Agosto, 2024
Tonantzintla, Puebla, México

Dirigida por:

Dra. María del Pilar Gómez Gil

Abstract

Convolutional Neural Networks (CNN) are one of the most used tools in the area of computer vision. When working with low-resolution images, its low memory cost per filter allows building deep architectures with a high level of accuracy. However, the deeper the architecture or the higher the resolution, the higher the computational cost. This is due to the convolution operation, which makes its implementation difficult, mainly in low-end devices. Several works have studied how to leverage the properties of the convolution theorem to reduce this cost, performing in the Fourier space (frequency representation). However, such proposal has drawbacks that hinders its implementation, for example, the definition of each basic CNN component in the frequency representation, including convolutional layers, activation functions and pooling layers. Furthermore, working in the frequency domain incurs in a considerable increase of the size of the filters, resulting in a greater demand for memory.

In this work, two alternatives are proposed to address these problems in the context of image classification, one focused on low-resolution images and the other for high-resolution images. In the first approach, two architectures are proposed, denominated type-R and type-LT; the type-R architectures work with a subset of the frequency components of the images, which we denominate reduced representation. The type-LT architectures, together with a novel layer denominated Linear Transform, dynamically build a reduced representation, which contains the most relevant information of the entire frequency representation. In the second approach, the Butterworth-CNN (BW-CNN) architecture is proposed, which uses all frequency components, with a mechanism with a small number of parameters, which generates Butterworth filters, and with a novel pooling layer called Spectral Average Pooling. The results using five data sets showed that the proposed architectures present substantial improvements, compared to other Fourier domain architectures in the state-of-the-art. Additionally, advantages over CNN in accuracy and computational cost were observed in some cases.

Resumen

Las redes neuronales convolucionales (CNN por sus siglas en inglés) son una de las herramientas más utilizadas en el área de visión por computadora. Cuando se manejan imágenes pequeñas, el bajo costo de memoria por filtro permite construir arquitecturas profundas con un alto nivel de exactitud. Sin embargo, a medida que se trabaja con arquitecturas más profundas o imágenes en alta resolución, el alto costo computacional de la operación de convolución dificulta su implementación, principalmente en dispositivos con recursos limitados. Varios trabajos han estudiado cómo aprovechar las cualidades del teorema de la convolución para aligerar este costo, operando en el espacio de Fourier (representación frecuencial). Sin embargo, estas propuestas presentan algunas dificultades en su implementación, como por ejemplo, la definición de todos los componentes básicos de una CNN en la representación frecuencial, incluyendo capas convolucionales, funciones de activación y capas de *pooling*. Además, trabajar en el dominio frecuencial conlleva un aumento considerable en el tamaño de los filtros, lo que resulta en una mayor demanda de memoria.

En este trabajo se proponen dos alternativas para afrontar estos inconvenientes en problemas de clasificación de imágenes, una enfocada en imágenes pequeñas y otra para imágenes grandes. En el primer enfoque, se proponen dos arquitecturas, llamadas tipo-R y tipo-LT; las arquitecturas tipo-R trabajan con un subconjunto de las componentes frecuenciales de las imágenes, lo que denominamos representación reducida. Las arquitecturas tipo-LT, en conjunto con una nueva capa denominada *Linear Transform*, construyen de manera dinámica una representación reducida que contiene la información más relevante de toda la representación frecuencial. Para el segundo enfoque, se propone la arquitectura Butterworth-CNN (BW-CNN), la cual usa todas las componentes frecuenciales, a través de un mecanismo con pocos parámetros que genera filtros tipo Butterworth, y una nueva capa de *pooling* denominada *Spectral Average Pooling*. Los resultados utilizando 5 bases de datos mostraron que las arquitecturas propuestas presentan mejoras sustanciales en comparación con otras arquitecturas en el dominio de Fourier encontradas en el estado del arte. Además, en algunos casos se observaron ventajas en la exactitud y costo computacional en contraste con CNN.

Agradecimientos

Quiero agradecer al INAOE por su respaldo durante la realización de este proyecto, así como a los profesores que generosamente compartieron su conocimiento conmigo y a los valiosos comentarios y sugerencias de mis sinodales. También quiero agradecer a mis amigos en el instituto por su apoyo, amistad y los buenos momentos compartidos.

Expreso mi sincero agradecimiento a la doctora María del Pilar Gómez Gil, mi asesora, cuyo apoyo, constante motivación y sabia orientación fueron fundamentales para la realización de esta investigación y contribuyeron significativamente a mi desarrollo académico.

Agradezco a mis padres y mi hermano por motivarme y apoyarme en esta etapa de mi vida, así como por enseñarme con su ejemplo el valor del esfuerzo y la dedicación.

A Mafer, te agradezco por tu cariño, comprensión y apoyo incondicional. Gracias por estar siempre a mi lado y la confianza que depositas en mí, la cual me impulsa a superarme día tras día.

Finalmente agradezco al Consejo Nacional de Humanidades, Ciencias y Tecnologías (CONAHCYT) por financiar esta investigación a través de la beca #1231141.

Gracias a todos,
Daniel.
Santa María Tonantzintla, Puebla, México.
Junio, 2024.

Tabla de Contenidos

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

Las redes neuronales convolucionales (CNN, por sus siglas en inglés) son una de las herramientas más usadas en el área de visión por computadora, principalmente en tareas de clasificación, detección o segmentación de imágenes y vídeos. Este tipo de redes aprovechan las cualidades de la operación de convolución, la cual, ha sido ampliamente usada en el área de procesamiento de señales digitales (DSP, por sus siglas en inglés), principalmente para el diseño de filtros y el tratamiento de imágenes y vídeo.

En el contexto de CNN, la operación de convolución se aplica en combinación con otro tipo de capas, comúnmente denominadas capas de *pooling*, y que tienen por objetivo reducir la dimensión de la representación. Además, las capas de convolución se evalúan usando funciones de activación no lineales. Este tipo de redes neuronales son ideales para trabajar con imágenes, ya que son capaces de identificar, hasta cierto grado, características complejas independientemente de la ubicación de éstas en la imagen. Además, su implementación normalmente utiliza un número reducido de parámetros, dado que cada *kernel*, esto es, la matriz que representa al filtro convolucional, frecuentemente se define usando dimensiones de 3×3 o 5×5, independientemente del tamaño de la representación de entrada, por lo cual, este tipo de redes ofrecen una opción eficiente en términos de memoria, comparadas con otros modelos de clasificación basados en redes neuronales.

Las cualidades de CNN permitieron la implementación de las primeras redes neuronales profundas, por ejemplo, la red neuronal profunda propuesta por el *Visual Geometric Group* , normalmente referida con las iniciales de éste (VGG) [2] o *Residual Network* (Resnet) [3]. Una importante cantidad de resultados experimentales con arquitecturas profundas indican que, a medida que una arquitectura contiene más capas convolucionales, aumenta la complejidad de las características que puede identificar y, por lo tanto, su exactitud. Sin embargo, en las redes neuronales profundas, la operación de convolución conlleva un alto costo computacional, principalmente cuando el tamaño de la representación de entrada es grande. Por consiguiente, la implementación de arquitecturas

convolucionales profundas requiere un costo computacional muy elevado, lo cual dificulta su aplicación en dispositivos con recursos limitados.

Algunos trabajos recientes proponen diversos mecanismos para aligerar el costo de la convolución. Por ejemplo, sugieren recurrir a otras operaciones para trabajar con imágenes [4], así como arquitecturas con distintas variantes de la convolución [5]. Otros enfoques buscan replicar las propiedades de la convolución sin depender directamente de esta operación [6]. Por otro lado, otras propuestas sugieren aprovechar las cualidades de la representación frecuencial de las imágenes. Esta representación, que contiene la misma información que la representación espacial, ofrece una serie de ventajas interesantes para reducir el costo computacional asociado con la convolución. Por otra parte, diversos autores proponen aprovechar las propiedades del teorema de la convolución para aligerar el costo computacional de CNN, sin perder las cualidades de la operación de convolución [7, 8, 9, 1, 10, 11, 12]. Este teorema establece que la convolución entre un par de señales en su representación espacial es equivalente al producto de Hadamard de la transformada de Fourier (representación frecuencial) de dichas señales [13]; esto resulta en un proceso computacionalmente menos costoso si se consideran algoritmos eficientes para calcular la transformada de Fourier, como el algoritmo *Fast Fourier Transform* (FFT) [14].

El teorema de la convolución ha sido utilizado en diversos trabajos de diseño de arquitecturas de redes neuronales, los cuales se basan en arquitecturas tipo CNN, pero recibiendo como entrada la representación frecuencial de las imágenes, lo que permite reemplazar la convolución por el producto de Hadamard. De aquí en adelante denominaremos a este enfoque "Redes de Fourier". Se ha observado que las Redes de Fourier muestran una reducción notable en el número de operaciones de punto flotante en comparación con CNN. Además, los *kernels* definidos en la representación frecuencial tienen la misma capacidad para extraer características que el enfoque tradicional. Sin embargo, este enfoque presenta una serie de inconvenientes que dificultan su implementación y aplicación, los cuales se describen en la siguiente sección y cuya solución son el tema de estudio de esta investigación.

1.1. Descripción del problema

El principal reto del enfoque de las Redes de Fourier, basado en reemplazar la operación de convolución por el producto de Hadamard, está en la definición del resto de los componentes de CNN usando una representación frecuencial, de tal manera que se mantenga la estructura de la red en esta representación, evitando

realizar múltiples transformaciones entre dominios.

Relacionado a este punto, se han propuesto diversas funciones de activación para trabajar con la representación frecuencial, por ejemplo [1, 10, 15, 11, 16, 17], sin embargo, ninguna ha mostrado un rendimiento significativamente superior a las demás. Con respecto a otros componentes, Rippel et al. [7] proponen una técnica de *pooling* que emula el mecanismo de filtros pasa-bajas, denominada *Spectral Pooling*. Esta técnica conserva las componentes frecuenciales bajas y desecha el resto. Trabajos posteriores reconocen esta capa como la más adecuada para trabajar en la representación frecuencial, ya que requiere un bajo costo computacional y demuestra resultados sobresalientes. Sin embargo, su mecanismo implica una pérdida considerable de información, lo que podría afectar la exactitud de las arquitecturas que emplean esta capa.

Por otro lado, trabajar en la representación frecuencial presenta un aumento considerable en el número de parámetros, ya que se utilizan *kernels* en variable compleja, con el mismo número de componentes que la imagen de entrada. Esto implica que se necesitan dos parámetros por cada componente del *kernel*, dando un total de $2 \times r \times c$ parámetros por *kernel*, donde $r \times c$ corresponde al tamaño de la representación. Este hecho dificulta la aplicación de este tipo de redes, ya que entre mayor sea el tamaño de la representación, mayor será el número de parámetros necesarios. Esta desventaja contrasta con CNN, cuyos *kernels* mantienen fijo el número de parámetros, comúnmente $k \times k$, con $k = 3$ o $k = 5$. Ayat et al. [1] sugieren una estrategia para afrontar este inconveniente, la cual consiste en definir únicamente los parámetros de los componentes que no serán eliminados por *Spectral Pooling*; sin embargo, su enfoque no consigue resolver completamente el problema en las primeras capas convolucionales. Del mismo modo, Han et al. [11] proponen una estrategia basada en *kernels* aleatorios caracterizados por dos parámetros entrenables. No obstante, los *kernels* aleatorios deben almacenarse como parámetros no entrenables, por lo cual aún representan un elevado costo de memoria.

Si bien las Redes de Fourier demuestran una reducción significativa en el costo computacional, aún existe un amplio margen de mejora, pues en los bloques fundamentales de CNN, en capas de *pooling* y en funciones de activación, aún se presentan deficiencias que obstaculizan el desempeño de este enfoque. Además, el problema del elevado número de parámetros dificulta su aplicación, ya que el enfoque convencional resulta más eficiente en este aspecto.

1.2. Solución propuesta

Como se mencionó anteriormente, las Redes de Fourier poseen la desventaja de requerir un elevado número de parámetros. Además, las técnicas propuestas hasta el momento ofrecen una oportunidad de mejora, particularmente la capa de *Spectral Pooling*, que presenta problemas de pérdida de información. Para afrontar estos inconvenientes, en este trabajo de investigación se proponen dos enfoques distintos:

- Enfoque para imágenes pequeñas: en este caso se busca trabajar con una representación reducida de las imágenes de entrada, compuesta por un conjunto reducido de sus componentes frecuenciales. Además, se elimina el uso de *Spectral Pooling* o cualquier otra técnica de *pooling*. Como resultado, la representación de entrada se mantiene con un tamaño reducido a través de toda la arquitectura, lo que resulta en una notable reducción en el número de parámetros y operaciones de punto flotante. Adicionalmente, se investiga de que manera construir dicha representación reducida con la información más indispensable de las componentes frecuenciales.

- Enfoque para imágenes grandes: en este enfoque se propone explorar otra alternativa para capas de *pooling*; el objetivo primordial es evitar la pérdida de información, lo cual es el principal inconveniente de *Spectral Pooling*. Además, se propone diseñar una función generadora de *kernels*, que dependa de un número fijo de parámetros, independientemente del tamaño de la representación de entrada, similar al enfoque que comúnmente se utiliza en arquitecturas CNN. De esta manera, la función generadora de *kernels* puede manejar entradas grandes.

Cabe mencionar que ambos enfoques tienen un propósito distinto. Mientras que el primero se adapta bien a imágenes pequeñas, donde es factible construir una representación reducida con un número limitado de componentes frecuenciales, esto no resulta viable con imágenes grandes. Similarmente, las características de la función generadora de *kernels* no son adecuadas para trabajar con imágenes pequeñas.

1.2.1. Objetivo general

El **objetivo general** de esta investigación es el diseñar e implementar arquitecturas de redes neuronales convolucionales ligeras en el espacio de Fourier, aprovechando las propiedades del teorema de la convolución.

1.2.2. Objetivos específicos

Los **objetivos específicos** son:

- Proponer alternativas para capas de *pooling* para la representación frecuencial.
- Diseñar estrategias para generar *kernels* en variable compleja con un número reducido de parámetros.
- Explorar enfoques distintos a los encontrados al estado del arte, incluyendo nuevas arquitecturas, mecanismos y capas.

1.3. Alcance y limitaciones

Las arquitecturas resultantes de esta investigación presentan una reducción significativa en el número de parámetros, en comparación con las Redes de Fourier propuestas en el estado del arte. Además, se logra una disminución en el número de operaciones de punto flotante en comparación con las Redes de Fourier y con las CNN convencionales, siendo más notable con respecto a éstas últimas. Asimismo, las estrategias propuestas aprovechan de mejor manera la información de la representación frecuencial, evitando pérdida de información debida a componentes frecuenciales no considerados. Esta característica se ve reflejada en una mayor exactitud en comparación con las Redes de Fourier propuestas en el estado del arte.

A pesar de estas ventajas, los resultados obtenidos no igualan el desempeño de CNN en todos los casos, principalmente al trabajar con conjuntos de imágenes naturales, como las encontradas en la base de datos CIFAR-10. Por otro lado, aunque las arquitecturas propuestas requieren significativamente menos operaciones que CNN, esta disminución no se ve reflejada en el tiempo de entrenamiento en todos los casos, principalmente debido a las diferencias en la optimización encontrada en las implementaciones de las redes CNN de código abierto, ya que nuestras capas se diseñaron considerando las clases de capas base proporcionadas por Keras, mientras que las capas disponibles en esta librería aprovechan otras herramientas para optimizar su ejecución.

1.4. Contribuciones

Las aportaciones realizadas en esta investigación son:

- En el enfoque para imágenes pequeñas:

- Arquitecturas tipo-R, las cuales reducen significativamente el número de parámetros y operaciones de punto flotante, en comparación con arquitecturas de Redes de Fourier propuestas en la literatura. Estas ventajas no comprometen el desempeño en clasificación de las arquitecturas y evitan el uso de capas de *Pooling*.
- Una nueva capa denominada *Linear Transform* capaz de identificar las componentes frecuenciales más significativas en una imagen. Esta capa forma parte de las arquitecturas también propuestas, denominadas arquitecturas tipo-LT, las cuales ofrecen los mismos beneficios que las arquitecturas tipo-R, pero muestran una mejora en la exactitud debido a las cualidades de la capa *Linear Transform*. Este incremento en la exactitud conlleva un ligero aumento en la cantidad de parámetros y operaciones de punto flotante en comparación con arquitecturas tipo-R.

- En el enfoque para imágenes grandes:
  - Una estrategia, denominada filtros Butterworth, diseñada para generar *kernels* de variable compleja, basada en el mecanismo de filtros pasa-bajas y pasa-altas ampliamente usados en el área de procesamiento de señales. Esta técnica muestra una notable reducción en el número de parámetros, ya que cada *kernel* se caracteriza por tener solo dos parámetros entrenables y dos no entrenables, independientemente de la dimensión de la representación de entrada.
  - Una nueva capa de *pooling* denominada *Spectral Average Pooling*, que busca aprovechar toda la información disponible en la representación frecuencial de las imágenes, abordando así el inconveniente de pérdida de información observado en las capas *Spectral Pooling*.

1.5. Organización del documento

El resto de este documento está organizado de la siguiente manera. En el Capítulo 2 se presentan los conceptos básicos que serán utilizados durante este trabajo. En el Capítulo 3 se revisan las propuestas más recientes de redes neuronales convolucionales en el espacio de Fourier, destacando sus ventajas y principales inconvenientes. El Capítulo 4 explica detalladamente los modelos propuestos durante esta investigación. El Capítulo 5 presenta los resultados experimentales obtenidos y una comparación con las técnicas más populares del estado del arte. Finalmente se muestran las conclusiones y el trabajo futuro en el Capítulo 6.

Capítulo 2

Marco teórico

En este capítulo se definen brevemente algunos de los conceptos que sirven de base para esta investigación. En primer lugar, se describen las cualidades de CNN. Posteriormente, se revisa la teoría de la transformada de Fourier, donde se incluye la definición de la transformada de Fourier centrada, así como la del teorema de la convolución, el cual es fundamental en los modelos utilizados en este trabajo.

2.1. Redes neuronales convolucionales

Las redes neuronales convolucionales son una de las herramientas más usadas en el área de visión por computadora. Su diseño está enfocado para trabajar con estructuras multidimensionales, por ejemplo: señales, imágenes, vídeo, etc. La efectividad de este tipo de redes quedó demostrada con la arquitectura propuesta por Krizhevsky et al. [18], denominada *AlexNet*. Esta arquitectura ganó el concurso *ImageNet object recognition challenge* en 2012, y es considerada una de las primeras redes profundas efectivamente entrenadas usando el modelo de *back propagation*, a pesar de que se creía que este enfoque de entrenamiento sería ineficiente.

CNN posee una motivación biológica, como lo indican los estudios de Hubel et al. [19] y de Fukushima [20], donde se muestra que el sistema de visión de los mamíferos reacciona a ciertos patrones de intensidad lumínica, los cuales, mediante estimulación de la retina, son procesados para ser interpretados por el cerebro. Análogamente, las arquitecturas CNN imitan este mecanismo mediante filtros convolucionales aprendidos durante el entrenamiento, los cuales buscan resaltar características especificas en la información de entrada, además, con el uso de múltiples capas convolucionales, es posible detectar características complejas. El sistema visual mamífero está organizado en una región bidimensional y es capaz de analizar la información que recibe gracias a un mecanismo formado por dos tipos de células: simples y complejas [19]. Las células simples son capaces de identificar patrones ubicados en regiones específicas, mientras que las células complejas son capaces de responder a patrones similares a aquellos caracterizados

por células simples. De esta manera los mamíferos son capaces de identificar objetos bajo distintas transformaciones: redimensionados, rotados, trasladados, etc. Este mecanismo es implementado en arquitecturas CNN a través de capas de *pooling*, las cuales reducen la dimensión de la representación de entrada, de tal manera que la diferencia entre una imagen y una copia ligeramente diferente es apenas perceptible.

La implementación de este tipo de arquitecturas no considera estrictamente la definición de la operación de convolución, sino la operación de correlación, ya que su cálculo es menos complicado. Este proceso resulta equivalente a la convolución con filtros opuestos por ambos vértices, por lo cual, las arquitecturas CNN poseen las cualidades de la operación de convolución, con el único inconveniente de definir filtros en una representación opuesta por sus vértices [21]. Las principales propiedades de estas arquitecturas son las siguientes [22]:

- *Sparse interactions*: a diferencia del Perceptron de Varios Niveles (MLP por sus siglas en inglés), el cálculo de cada componente del mapa de características de salida de una capa convolucional no depende de todas las componentes del mapa de características de entrada. Dependiendo del tamaño del filtro convolucional, solo cierta región de componentes es considera en el cálculo.

- *Parameter sharing*: esta característica permite que cada parámetro en un filtro convolucional sea usado en repetidas ocasiones por distintos componentes del mapa de características de entrada. De esta manera, sólo es necesario aprender un conjunto reducido de parámetros para toda la representación de entrada, en lugar de aprender un conjunto de parámetros por cada componente, como en el caso de MLP. Esta característica permite diseñar arquitecturas con un número de parámetros entrenables reducido.

- *Equivariance to translations*: el proceso de convolucionar una imagen o de trasladarla es un caso de transformaciones equivariantes, es decir, resulta equivalente convolucionar una imagen y posteriormente desplazarla, que realizar el proceso inverso. De esta manera, las características de interés resaltadas por un filtro serán procesadas de la misma manera independientemente de su ubicación en la representación de entrada.

2.1.1. Funciones de activación

Las funciones de activación en CNN no poseen una motivación biológica directa como en el caso de MLP, donde cada neurona es evaluada con una función de activación no lineal para simular la estimulación que perciben las

neuronas al recibir impulsos eléctricos de distintas magnitudes [21]. Sin embargo, desde el punto de vista matemático, las funciones de activación no lineales permiten aproximar una gran variedad de funciones continuas, por lo cual CNN es capaz de aproximar comportamientos complicados al trabajar con funciones de activación.

La función sigmoide $\sigma(x)$ es una de las funciones de activación más usadas para trabajar con redes neuronales. Esta función permite limitar los valores de salida dentro del rango de 0 a 1, estados a los que se asocia una neurona completamente apagada y otra completamente activada, respectivamente. $\sigma(x)$ se define en la Ecuación 2.1.

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.1}$$

La función $\mathtt{sigmoide}$ suele utilizarse como activación de cualquier neurona y en clasificación binaria. En este contexto se asigna la etiqueta 0 a los valores por debajo de 0,5 y la etiqueta 1 a los valores superiores a 0,5. Por otro lado, la función de activación $\mathtt{softmax}$ suele usarse para problemas de clasificación con más de dos clases. Esta función se define en la Ecuación 2.2:

$$\mathtt{softmax}(x_i) = \frac{exp(x_i)}{\sum_j exp(x_j)} \tag{2.2}$$

donde x_i representa la salida i-ésima de la capa de clasificación y $\mathtt{softmax}(x_i)$ representa la probabilidad de que la instancia a clasificar pertenezca a la clase i-ésima.

Por otro lado, ReLU (*Rectified Linear Unit*) es una función de activación no lineal con una definición simple y poco costosa computacionalmente hablando (ver Ecuación 2.3). Al igual que la función sigmoide, ReLU tiene una motivación biológica, ya que se activa cuando su entrada supera el valor cero o un valor umbral. Esta función es utilizada en casi todas las capas de una red neuronal, excepto en capas de clasificación. A diferencia de sigmoide, ReLU presenta una definición menos costosa computacionalmente hablando y tiende a converger hacia resultados óptimos en menos iteraciones que otras funciones.

$$\mathtt{ReLU}(x) = \mathtt{max}\{0, x\} \tag{2.3}$$

2.1.2. Capas de *pooling*

Como se mencionó anteriormente, el mecanismo de una capa de *pooling* consiste en reducir el tamaño de la imagen de entrada. Esto se realiza para eliminar el ruido en las imágenes y así evitar que la detección de características dependa de su ubicación en la imagen. Además, se disminuye el número de operaciones en capas posteriores al reducir el tamaño de la representación.

La Figura 2.1 presenta dos técnicas de *pooling* ampliamente usadas en arquitecturas CNN: *Max pooling* y *Average Pooling*. Ambas capas se implementan dividiendo la imagen de entrada en ventanas de un tamaño y posición dadas por la persona usuaria. A estas ventanas se les aplica una transformación para obtener un único valor con la información de cada ventana. En el caso de *Max Pooling*, se elige el valor máximo en cada ventana, mientras que *Average Pooling* calcula el promedio de todos los valores en cada ventana.

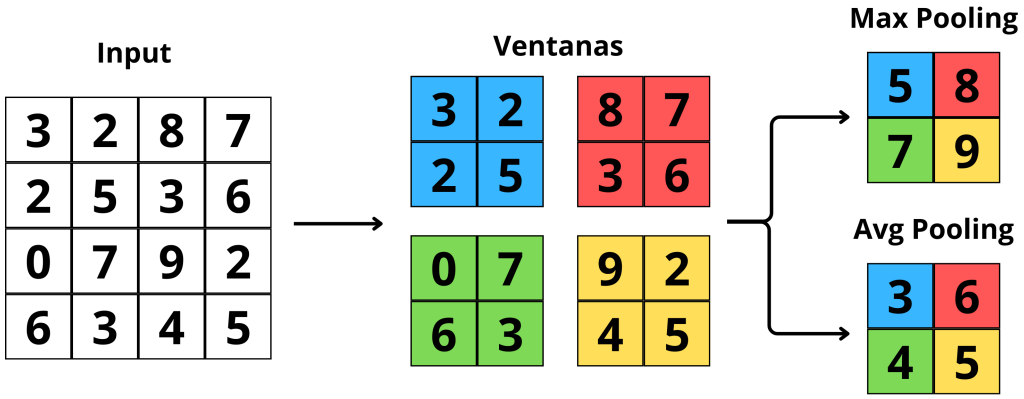


Figura 2.1: Mecanismos de las capas *Max pooling* y *Average Pooling*, seleccionando el valor máximo y promedio de las ventanas, respectivamente.

2.2. Procesamiento de señales digitales

En esta sección se revisan conceptos fundamentales del área de procesamiento de señales digitales, comenzando con la definición de una señal, su transformada de Fourier y transformada de Fourier centrada. Además, se incluye una breve revisión del teorema de la convolución, ampliamente usado en esta área para aligerar el costo computacional de la convolución.

2.2.1. Transformada de Fourier

La Transformada de Fourier es una transformación matemática cuyas propiedades han sido ampliamente aprovechadas en el área de procesamiento de señales. Esta transformación se caracteriza por estudiar señales espaciales en términos de una serie armónica.

En nuestro caso, nos interesa el estudio de señales discretas. Definiremos a este tipo de señales x como un conjunto de N valores complejos x_i, como se muestra en la Ecuación 2.4.

$$x = \{x_0, x_1, \ldots, x_{N-1}\} \tag{2.4}$$

La hipótesis base de Fourier indica que cualquier señal discreta x, puede escribirse en términos de una serie armónica [23], tal y como se muestra en la Ecuación 2.5.

$$x_j = \sum_{n=0}^{N-1} X_n \exp\left(2\pi i \frac{nj}{N}\right) \tag{2.5}$$

El conjunto de valores X_n se denomina representación frecuencial de x, denotada como $X = \mathcal{F}(x) = \{X_0, X_1, X_2, \ldots, X_{N-1}\}$. Análogamente, X también puede expresarse de manera similar:

$$X_j = \sum_{n=0}^{N-1} x_n \exp\left(-2\pi i \frac{nj}{N}\right) \tag{2.6}$$

Las Ecuaciones 2.6 y 2.5 reciben el nombre de transformada de Fourier y transformada de Fourier inversa, respectivamente. Se dice que x es una señal en la representación espacial, mientras que X es una señal en la representación frecuencial [13].

En el contexto de visión por computadora, el cual centra su estudio en el análisis de imágenes, las señales se caracterizan por una representación discreta de dos dimensiones $x_{j,k}$. Análogo al caso en una dimensión, se definen la transformada de Fourier y la transformada de Fourier inversa en dos dimensiones en las Ecuaciones 2.8 y 2.7:

$$x_{j,k} = \frac{1}{\sqrt{MN}} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} X_{m,n} \exp\left[2\pi i \left(\frac{mj}{M} + \frac{nk}{N}\right)\right] \tag{2.7}$$

$$X_{j,k} = \frac{1}{\sqrt{MN}} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x_{m,n} \exp\left[-2\pi i \left(\frac{mj}{M} + \frac{nk}{N}\right)\right] \tag{2.8}$$

donde x y X son señales discretas de dimensión $M \times N$ [23].

2.2.2. Teorema de la convolución

El teorema de la convolución establece una relación entre la operación de convolución y el producto de Hadamard (producto punto a punto) [13]. Este teorema representa uno de los resultados más sobresalientes del área de procesamiento de señales, dado que permite reducir el costo computacional de la operación de convolución [23].

Dadas un par de señales x y y en la representación espacial, y sus respectivas transformadas de Fourier X y Y, el teorema de la convolución establece que la convolución de x y y es equivalente al producto de Hadamard de X y Y.

Análogamente, el producto de Hadamard de x y y es equivalente a la convolución de X y Y. Tal y como se aprecia en las Ecuaciones 2.9 y 2.10.

$$x \circledast y = \mathcal{F}^{-1}(X \odot Y) \tag{2.9}$$

$$x \odot y = \mathcal{F}^{-1}(X \circledast Y) \tag{2.10}$$

donde $\circledast$ y $\odot$ representan la operación de convolución y el producto de Hadamard, respectivamente, y $\mathcal{F}^{-1}$ denota a la transforma de Fourier inversa.

Estos resultados permiten reducir el elevado costo computacional asociado al cálculo de la convolución, como lo sugiere la Ecuación 2.9, dado que el producto de Hadamard es menos costoso que la operación de convolución. Además, si se consideran algoritmos eficientes para calcular la transformada de Fourier, como el algoritmo *Fast Fourier Transform* (FFT) [14], entonces el costo computacional reduce aún más. Por ejemplo, dadas un par de señales uni-dimensionales de longitud N, el cálculo de su transformada de Fourier es de complejidad $O(N \log N)$, su producto de Hadamard posee una complejidad $O(N)$, y la transformada de Fourier inversa de dicho resultado es de complejidad $O(N \log N)$. Entonces, la complejidad de todo este proceso es $O(N \log N)$. En contraste, el cálculo de la convolución de las mismas señales es de complejidad $O(N^2)$. Por esta razón resulta más conveniente trabajar en la representación frecuencial de las señales.

2.2.3. Transformada de Fourier centrada

La definición de la transformada de Fourier coloca a las componentes de frecuencia baja y alta dispersas en distintas posiciones de la representación frecuencial. Por esta razón, es común trabajar con la transformada de Fourier centrada, la cual organiza las componentes frecuenciales de manera intuitiva. Dada una señal en su representación espacial x y su correspondiente representación frecuencial X, ambas señales de dimensión $M \times N$, la transformada de Fourier centrada se calcula con la siguiente transformación de coordenadas:

$$j' = j - \frac{M}{2}; \qquad k' = k - \frac{N}{2} \tag{2.11}$$

Al realizar este cambio de coordenadas, se observa que la transformada de Fourier centrada X', denominada representacion frecuencial centrada, es equivalente a la transformada de Fourier de la señal x desplazada en la representación espacial, tal y como se muestra en la Ecuación 2.12.

$$X'_{j',k'} = \mathcal{F}\left[(-1)^{j+k} x_{j,k}\right] \tag{2.12}$$

Otra manera equivalente para calcular X' consiste en dividir la representación X en cuadrantes e intercambiar los pares opuestos por las esquinas [23]. La Figura 2.2 muestra un diagrama de como se realiza dicho proceso.

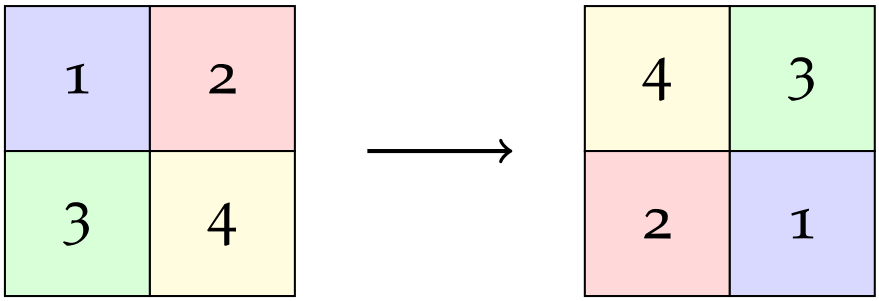


Figura 2.2: Proceso de conversión para obtener una representación frecuencial centrada de la transformada de Fourier

Esta representación resulta más conveniente que la transformada de Fourier convencional, ya que reorganiza las componentes de frecuencia alta a las extremos de la representación, mientras que las componentes frecuenciales bajas son ubicadas en la parte central de la representación, con la componente X_{00} ubicada exactamente en el centro. Esta componente recibe el nombre de componente DC (*direct current*), haciendo a alusión a una señal eléctrica estática de frecuencia cero. Este enfoque facilita el diseño de filtros, tal y como se explica en la siguiente sección.

2.3. Filtrado de imágenes en la representación frecuencial

En el área de procesamiento de señales, es común trabajar con la representación frecuencial centrada de las imágenes para aplicar filtros.

El proceso de filtrado de imágenes en el dominio espacial consiste en convolucionar una imagen con un filtro diseñado especialmente para resaltar o eliminar características específicas de la imagen, como bordes, texturas, etc. El proceso es similar en la representación frecuencial, sin embargo, existen ciertas ventajas en este dominio. En primer lugar, el calcular la convolución en la representación frecuencial presenta una reducción considerable en el número de operaciones (ver Sección 2.2.2). Por otro lado, resulta más intuitivo diseñar filtros con ayuda de la representación frecuencial centrada (ver Sección 2.2.3).

El proceso de filtrar imágenes en la representación frecuencial consiste en resaltar o atenuar ciertas componentes de la representación frecuencial centrada de las imágenes. Por ejemplo, los filtro pasa-bajas permiten el paso de las componentes frecuenciales bajas ubicadas en el centro de la representación frecuencial centrada, mientras que el resto son simplemente eliminadas. Por el contrario, un filtro pasa-altas elimina las componentes ubicadas en el centro

de la representación y preserva el resto de componentes, las cuales representan las componentes frecuenciales altas [24]. Las Ecuaciones 2.13 y 2.14 definen los filtros ideales pasa-bajas y pasa-altas, respectivamente:

$$H(u,v) = \begin{cases} 1 \text{ si } (u,v) \in \Delta_x \\ 0 \text{ si } (u,v) \notin \Delta_x \end{cases} \tag{2.13}$$

$$H(u,v) = \begin{cases} 0 \text{ si } (u,v) \in \Delta_x \\ 1 \text{ si } (u,v) \notin \Delta_x \end{cases} \tag{2.14}$$

donde Δ_x representa una región centrada de componentes con dimensión dada por la persona usuaria. Cuando Δ_x no se encuentra centrada, entonces los filtros definidos por las Ecuaciones 2.13 y 2.14 se denominan pasa-bandas y rechaza-bandas, respectivamente [23]. Estos filtros actúan sobre la representación frecuencial centrada por medio del producto de Hadamard, de esta manera, las componentes multiplicadas por 1 son preservadas, mientras que aquellas que son multiplicados por 0 son eliminadas.

Cada filtro posee propiedades características, por ejemplo, los filtros pasa-bajas difuminan la representación de entrada, dado que las componentes frecuenciales bajas contienen información de las texturas de la imagen. Por otro lado, los filtros pasa-altas resaltan los bordes la imagen, ya que las componentes frecuenciales altas representan cambios abruptos entre píxeles, provocados principalmente por los bordes de los objetos en la imagen. Por otra parte, los filtros pasa-bandas son utilizados para analizar información localizada en distintas componentes frecuenciales, y los filtros rechaza-bandas son utilizados para eliminar ruido provocado por componentes frecuenciales especificas. La Figura 2.3 ilustra la reconstrucción resultante de aplicar cada uno de estos filtros a una imagen.

Este enfoque presenta diversas ventajas en comparación con el proceso de filtrado en la representación espacial. Sin embargo, su aplicación presenta ciertas dificultades. Por ejemplo, los filtros pasa-bandas y rechaza-bandas requieren de una búsqueda exhaustiva para identificar la posición más adecuada para colocar la región Δ_x [23].

2.4. Métricas de clasificación

En esta sección se revisa brevemente la definición de un clasificador, así como las principales métricas utilizadas para evaluar su desempeño al trabajar sobre un conjunto de datos determinado.

El problema de clasificación supervisada de instancias consiste en proponer un clasificador capaz de predecir la clase de cada instancia. Una definición

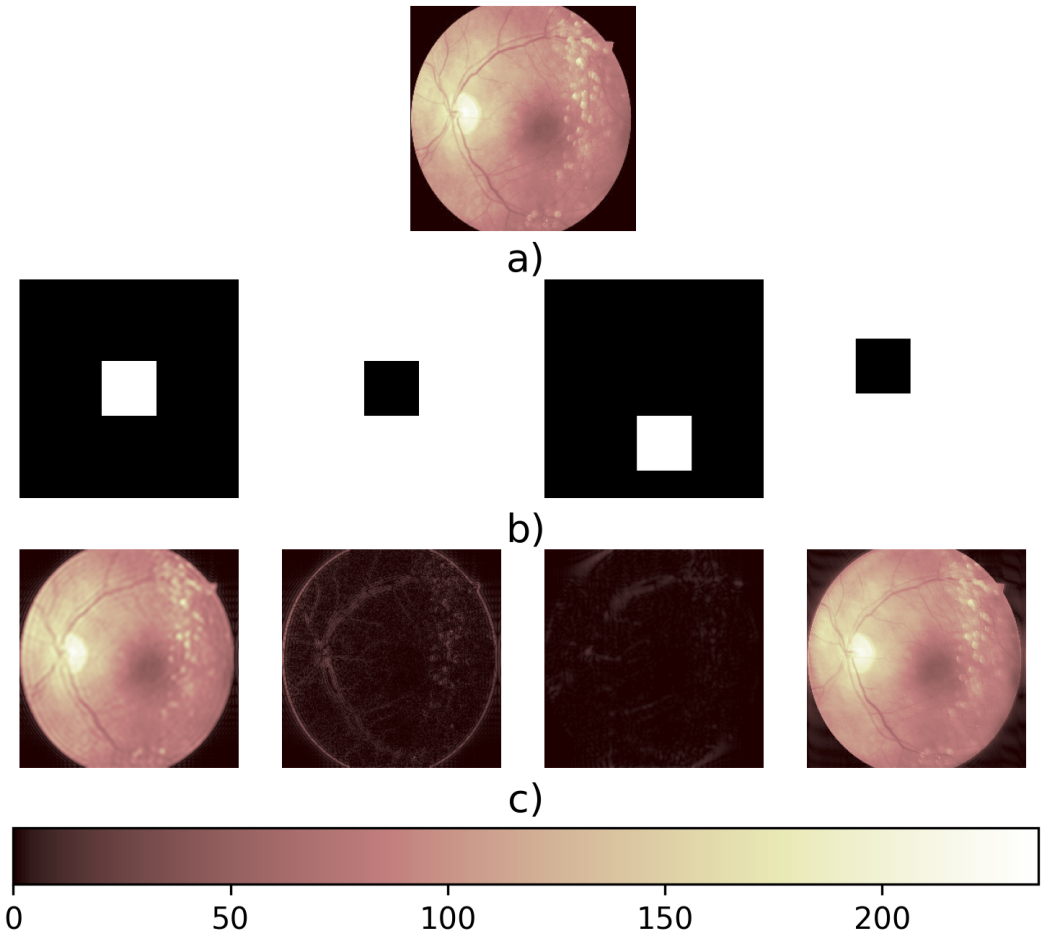


Figura 2.3: Ejemplos de imágenes filtradas. a) Imagen original. b) *Kernels* pasa-bajas, pasa-altas, pasa-bandas y rechaza-bandas, respectivamente. c) Resultado de evaluar la imagen original con el *kernel* ubicado en la parte superior

más formal es la siguiente: dado un conjunto de N duplas definido como $\{(X_1, y_1), (X_2, y_3), \ldots, (X_N, y_N)\}$, donde X_i representa la información de una instancia en particular y y_i la clase a la cual pertenece X_i, el objetivo de un clasificador es proponer una función f tal que $f(X_i) = y_i$, para todo X_i [25]. Sin embargo, en la práctica es complicado definir con exactitud a la función f, por lo cual la estrategia a seguir es encontrar una aproximación a dicha función. Para ello es necesario contar con métricas para medir el desempeño del clasificador.

Dado un problema de clasificación binario, donde los únicos valores posibles para la clase de cada instancia son positivo o negativo, denotados por c^+ y c^-, respectivamente, y un clasificador aplicado a este problema, dependiendo de la etiqueta real de cada instancia y la predicción del clasificador a evaluar, se tienen los casos: positivo verdadero (TP, *True Positive*), positivo falso (FP, *False Positive*), negativo verdadero (TN, *True Negative*) y negativo falso (FN, *False Negative*). Tal y como se muestra en la Tabla 2.1.

Tabla 2.1: Combinaciones posibles entre la clase real de una instancia y la predicción de un clasificador

		Predicción	
		c^+	c^-
Real	c^+	TP	FN
	c^-	FP	TN

Dado un experimento, en el cual se pone a prueba al clasificador evaluándolo con un conjunto de prueba, se define la exactitud como el número de predicciones correctas dividido entre el número total de predicciones (ver Ecuación 2.15).

$$\text{exactitud} = \frac{TP + TF}{TP + TF + FP + FN} \tag{2.15}$$

Note que esta métrica busca minimizar el total de errores, ya que cuando FP y FN es igual a cero, entonces la exactitud alcanza el valor máximo. Esta medida resulta intuitiva en la práctica, ya que se interpreta como el porcentaje de predicciones acertadas; sin embargo, presenta inconvenientes al trabajar con clases desbalanceadas, es decir, cuando el número de instancias de cada clase en el conjunto a evaluar presentan una diferencia significativa. En este caso el clasificador tiende a predecir la clase mayoritaria, pues es probable que acierte [25].

Existen otras medidas de evaluación que afrontan este inconveniente, por ejemplo la precisión y el recuerdo, expresadas en las Ecuaciones 2.16 y 2.17 [26], respectivamente.

$$\text{precisión} = \frac{TP}{TP + FP} \tag{2.16}$$

$$\text{recuerdo} = \frac{TP}{TP + FN} \tag{2.17}$$

La precisión busca minimizar el número de positivos falsos, al medir la porción de instancias clasificadas como positivas por el clasificador que realmente son positivas. Por otro lado, el recuerdo minimiza el número de negativos falsos, en este caso se mide la porción de instancias positivas que fueron clasificadas como positivas por el clasificador.

Otra métrica ampliamente usada en la literatura es F1-Score (Ecuación 2.18), la cual combina la información de la precisión y el recuerdo a través de una media armónica.

$$\text{F1-Score} = 2 \times \frac{\text{precisión} \times \text{recuerdo}}{\text{precisión} + \text{recuerdo}} \tag{2.18}$$

La definición de estas métricas puede extenderse a problemas de clasificación multi-clase [26]. En primer lugar, para la exactitud, la Ecuación 2.15 puede

extenderse a la siguiente definición:

$$exactitud = \frac{\text{\# predicciones correctas}}{\text{\# instancias}} \tag{2.19}$$

Por otro lado, para las métricas precisión, recuerdo y F1-Score, se suelen utilizar dos estrategias. La primera, evalúa estas métricas por separado para cada clase, considerando como positiva a la clase en cuestión y al resto de clases como negativas. Por otro lado, en el segundo enfoque se considera el promedio de las métricas individuales de cada clase.

En resumen, las arquitecturas CNN, al trabajar con la operación de convolución, presentan cualidades adecuadas para trabajar con imágenes, además, su implementación requiere de un menor consumo de memoria en comparación con otras opciones. Similarmente, en el área de DSP, la operación de convolución es ampliamente usada, abordando su alto costo computacional al considerar al teorema de la convolución. En este enfoque, al trabajar con imágenes en la representación frecuencial, se reduce significativamente el costo computacional de la convolución, además, el diseño de filtros convolucionales resulta más intuitivo en esta representación.

Capítulo 3

Trabajo relacionado

En este capítulo se describen los aportes más recientes a redes neuronales convolucionales. En primer lugar se describen extensiones a CNN en la representación espacial, posteriormente se revisan propuestas que aprovechan las propiedades de la transformada de Fourier para implementar redes neuronales convolucionales en la representación frecuencial de imágenes. Adicionalmente, se dedican un par de secciones para describir aportes y modelos que serán usados como base para el método propuesto.

3.1. Aportes a CNN

Las redes neuronales convolucionales son una de las herramientas más utilizadas actualmente en el área de visión por computadora. Originalmente, el mecanismo de esta arquitectura fue propuesto por Fukushima [20] con un modelo de aprendizaje no supervisado. Posteriormente, Lecun et al. [27], aportan en esta área proponiendo un aprendizaje supervisado basado en gradiente. Con el paso del tiempo, diversas arquitecturas basadas en CNN han sido propuestas, consiguiendo resultados sobresalientes en diversas tareas como clasificación, detección o segmentación de imágenes.

Recientemente se han desarrollado diversas alternativas para sustituir o mejorar las características y el desempeño de CNN. Por ejemplo, Li et al. [4] y Bert et al. [28], proponen soluciones basadas en filtros generados dinámicamente en función de las entradas de la red; de esta manera, consiguen arquitecturas que aprovechan las características espaciales de las imágenes. Por otro lado, Hu et al. [29], introducen un bloque denominado *Squeeze-and-Excitation Block*, el cual resalta de manera dinámica las características extraídas por cualquier transformación; de este modo se puede implementar este bloque en cualquier arquitectura. En particular, estos autores consiguen resultados sobresalientes al implementar dichos bloques con capas convolucionales. Estas técnicas resultan en arquitecturas con un desempeño superior al conseguido únicamente con capas convolucionales; sin embargo, al mismo tiempo implican un costo computacional adicional. La

tendencia indica que, entre mejores sean los resultados obtenidos, mayor será el costo computacional de las arquitecturas, tanto para el entrenamiento como para la inferencia.

Por otro lado, se han propuesto diversas alternativas que buscan implementar arquitecturas ligeras sin perder las cualidades de la convolución. Tolstikhin et al. [6] introducen una arquitectura basada en perceptrón multi-capa (MLP por sus siglas en inglés), cuya estructura imita las características de la operación de convolución. Por otro lado, Zamora et al. [30], abordan la idea de *kernels* adaptativos junto con la operación de convolución tradicional; sus arquitecturas consiguen resultados sobresalientes, reduciendo el número de *kernels* y de capas, en comparación con arquitecturas convencionales. Otro aporte reciente es el modelo *MobileNet*, propuesto por Howard et al. [5]. Esta propuesta se enfoca en aplicaciones para dispositivos inteligentes con pocos recursos computacionales, aprovecha las capacidades de convoluciones *depth-wise* separables, consiguiendo arquitecturas ligeras para aplicaciones enfocadas a dispositivos móviles.

3.2. Redes neuronales convolucionales en el espacio de Fourier

Recientemente se han propuesto diversas alternativas para extraer información de imágenes y proponer arquitecturas de redes neuronales ligeras aprovechando las propiedades del área de procesamiento de señales, en particular, de las cualidades de la transformada de Fourier, ya que esta representación contiene la misma información que la representación espacial y posee ciertas características que permiten aligerar cálculos.

Trabajos recientes han estudiado la manera de aprovechar el teorema de convolución para implementar arquitecturas con las características de CNN pero realizando el cálculo de la convolución en la representación frecuencial; nos referiremos a este tipo de propuestas como Redes de Fourier. Varios trabajos resaltan en este contexto. Por ejemplo, Mathiew et al. [31] y Lin et al. [32] aprovechan el teorema de la convolución para construir arquitecturas tipo CNN que realizan el cálculo de la operación de convolución en el espacio de frecuencias, reduciendo el costo computacional tanto del entrenamiento como de la inferencia, comparado con CNN. Sin embargo, estas propuestas presentan el inconveniente de realizar constantes transformaciones entre la representación espacial y la representación frecuencial, ya que por cada convolución son necesarias las transformadas de Fourier de la entrada y sus *kernels*; posteriormente se calcula su producto de Hadamard y finalmente se requiere calcular la transformada de

Fourier inversa del resultado. Una vez que se obtiene una representación espacial, se utilizan capas de *pooling* y funciones de activación convencionales. Ante esta situación, resulta evidente la necesidad de definir el resto de componentes de una CNN en la representación frecuencial, incluyendo funciones de activación y capas de *pooling*. Por consiguiente, se mantiene la extracción de características en la representación frecuencial y se reduce aún más el costo computacional.

Las propuestas de Pratt et al. [8], Ayat et al. [1] y Han et al. [11] imitan la estructura de una CNN convencional, adaptando su mecanismo para trabajar con la representación frecuencial de las imágenes. En estos enfoques se proponen distintas funciones de activación para números complejos, aprovechando las cualidades la representación compleja rectangular y polar. Además, se aprovecha la capa de *pooling* propuesta por Rippel et al. [7], denominada *Spectral Pooling*. Por otro lado, Watanabe et al. [10] y Zak et al. [12] proponen arquitecturas con un enfoque distinto a CNN. En el trabajo de Watanabe et al. [10] se aprovecha por separado la información de cada cuadrante y el centro de la representación frecuencial, mientras que la propuesta de Zak et al. [12] trabaja exclusivamente con la parte real de la representación frecuencial. En la Tabla 3.1 se comparan algunas de las propuestas en el estado del arte para trabajar en espacio de Fourier.

3.3. *Spectral Pooling*

En el estado del arte, *Spectral Pooling* ha sido reconocida como la técnica de *pooling* más adecuada para trabajar con Redes de Fourier. Originalmente fue propuesta por Rippel et al. [7] y se ha aprovechado de distintas maneras en trabajos posteriores.

El objetivo principal de las capas de *pooling* en redes neuronales convolucionales es reducir la dimensionalidad de la representación; de esta manera evitamos que el aprendizaje de la red dependa de características espaciales particulares a la imagen. En el caso de *Spectral Pooling* se propone un mecanismo para reducir la dimensionalidad de la representación, permitiendo el paso de ciertas frecuencias y eliminando al resto de los componentes.

Dada una señal de entrada X de dimensión $M \times N$ en su representación frecuencial centrada (ver Sección 2.2.3), y una dimensión de salida esperada $H \times W$, *Spectral Pooling* permite el paso de las frecuencias cubiertas por una ventana centrada de dimensión $H \times W$; el resto de frecuencias simplemente se eliminan. De esta manera se consigue una representación de dimensión $H \times W$, la cual contiene únicamente la información de la región central de la representación

Tabla 3.1: Comparación de técnicas del estado del arte. Los espacios en blanco indican que la propuesta no usa técnicas en la representación frecuencial

Autores	Representación	*Kernels*	Activación	*Pooling*
2015-Rippel et al. [7]	Compleja rectangular (espacial-frecuencial centrada)	Representación frecuencial		*Spectral Pooling*
2017-Pratt et al. [8]	Compleja rectangular (frecuencial centrada)	Representación frecuencial.		*Spectral Pooling*
2018-Trabelsi et al. [9]	Espacial	Representación frecuencial	ZReLU	
2019-Ayat et al. [1]	Compleja rectangular (frecuencial-espacial)	Representación frecuencial	SReLU	*Spectral Pooling* (sobre primer cuadrante)
2020-Watanabe et al. [10]	Compleja polar (frecuencial centrada)	Representación frecuencial por cuadrantes	2SReLU	*Spectral Pooling*
2021-Han et al. [11]	Compleja rectangular-polar (frecuencial centrada)	Representación espacial y *kernels* aleatorios en representación frecuencial	PhaseReLU	*Spectral Pooling*
2023-Zack et al. [12]	Compleja rectangular (frecuencial-espacial)	Representación frecuencial (parte real)		

de entrada. La Figura 3.1 muestra un diagrama del funcionamiento de esta técnica, aplicada a una entrada en su representación frecuencial centrada. Note que, la motivación de esta técnica proviene del área de procesamiento de señales y se puede interpretar como un filtro pasa-bajas, ya que preserva únicamente una región de componentes frecuenciales bajas, similar a la Ecuación 2.13.

Spectral Pooling ha demostrado ser una técnica con resultados sobresalientes, incluso en comparación con otras opciones que trabajan con la representación espacial, ya que esta técnica muestra una menor pérdida en los detalles de una imagen, tal y como se aprecia en la Figura 3.2, donde se compara la pérdida de detalles de *Max Pooling* (**MaxPool**) y *Spectral Pooling* (**SPool**), adicionalmente se muestra otra variante de *Spectral Pooling* (**SPool-Ayat**), la cual se enfoca

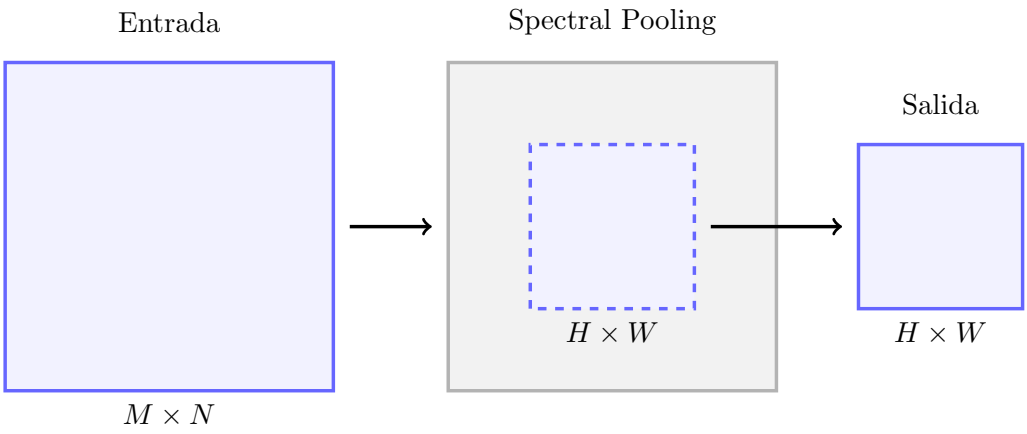


Figura 3.1: Descripción del funcionamiento de la capa *Spectral Pooling*

en las componentes del primer cuadrante de la representación frecuencial no centrada; esta última modificación fue propuesta por Ayat et al. [1]. La gran capacidad de *Spectral Pooling* de preservar detalles se atribuye a que la mayor parte de información estructural de una imagen se concentra en las componentes frecuenciales bajas.

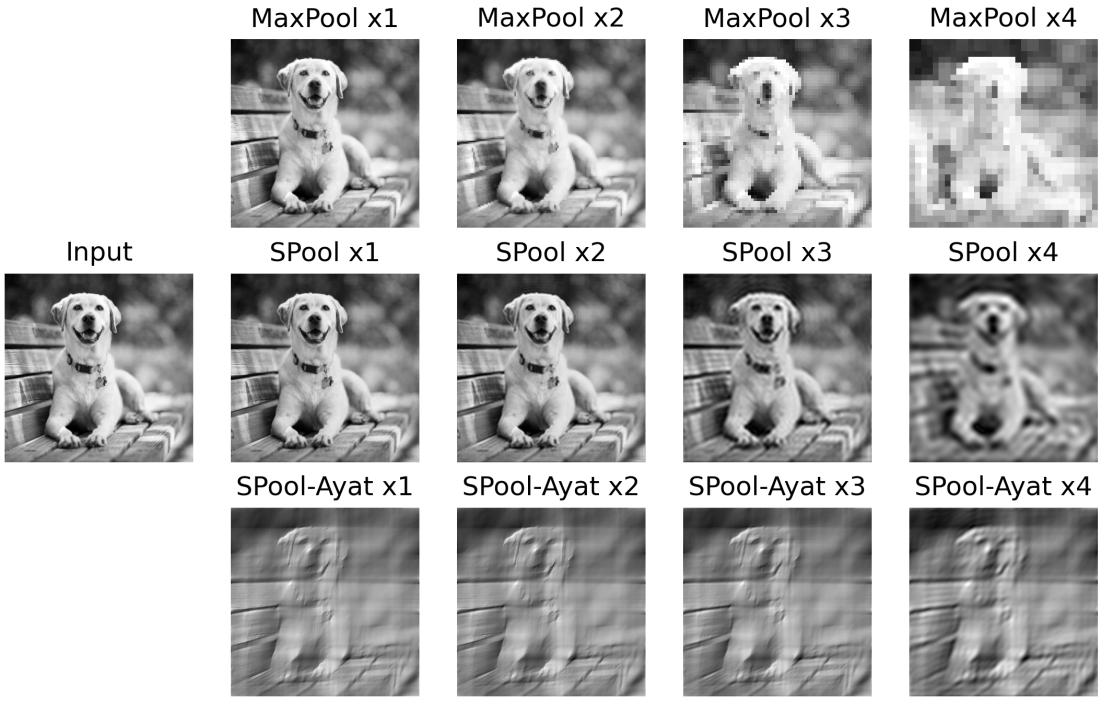


Figura 3.2: Comparación de técnicas de *pooling*: *Max Pooling* (MaxPool), *Spectral Pooling* (SPool) y una variante de *Spectral Pooling* propuesta por Ayat et al. [1] (SPool-Ayat). Los ejemplos se muestran con distintos grados de reducción

3.4. Funciones de activación en la representación frecuencial

Como ya se mencionó anteriormente, es necesario definir todos los bloques fundamentales de CNN en la representación frecuencial. En el caso de las funciones de activación, es evidente la necesidad de nuevas opciones, ya que cada componente de la representación frecuencial es un número complejo, a diferencia del enfoque convencional, en el cual las funciones de activación están definidas para números reales.

Ayat et al. [1] proponen la función de activación SReLU, la cual utiliza la operación de convolución para replicar el comportamiento de ReLU (ver Sección 2.1.1) en la representación espacial. SReLU se define como sigue:

$$\mathrm{SReLU}(z) = \mathrm{DC} + c_1 \cdot z + c_2 \cdot z \circledast z \tag{3.1}$$

donde $z \in \mathbb{C}$, $\mathrm{DC} \to 0$ y los parámetros c_1 y c_2 se ajustan para imitar el comportamiento de ReLU en cierto rango. De forma similar, Watanabe et al. [10] proponen una variante a SReLU, denominada 2SReLU. A diferencia de su predecesor, esta función de activación incorpora información de los segundos armónicos de la representación frecuencial ($F(\mu_2)$) a los componentes de frecuencia baja $F(\mu_1)$ y elimina el uso de la operación de convolución. 2SReLU se define con la transformación descrita en la Ecuación 3.2:

$$F(\mu_1) \to \alpha F(\mu_1) + \beta F(\mu_2) \tag{3.2}$$

donde las constantes α y β determinan la contribución de cada término.

Por otra parte, Trabelsi et al. [15] proponen la función de activación denominada CReLU. Su trabajo propone un enfoque diferente, en el cual, se extiende la definición convencional de la función de activación ReLU al dominio complejo. Su propuesta consiste en evaluar por separado la parte real e imaginaria de la entrada, como se define en la Ecuación 3.3:

$$\mathrm{CReLU}(z) = \mathrm{ReLU}(\mathcal{R}(z)) + i\,\mathrm{ReLU}(\mathcal{I}(z)) \tag{3.3}$$

donde $\mathcal{R}(z)$ y $\mathcal{I}(z)$ representan la parte real e imaginaria de z, respectivamente.

Por otro lado, Han et al. [11], Arjovsky et al. [16] y Guberman et al. [17] proponen las función de activación PhaseReLU, modReLU y zReLU, respectivamente. La definición de estas funciones de activación involucra la información de la fase y norma de la representación frecuencial, tal y como se aprecia en las Ecuaciones 3.4, 3.5 y 3.6:

$$\mathrm{PhaseReLU}(z) = |z|\,[\cos(\mathrm{ReLU}(\theta)) + i\sin(\mathrm{ReLU}(\theta))] \tag{3.4}$$

$$\text{modReLU}(z) = \text{ReLU}(|z| + b)\, e^{i\theta} \tag{3.5}$$

$$\text{zReLU}(z) = \begin{cases} z & \text{si } \theta \in \left[0, \frac{\pi}{2}\right] \\ 0 & \text{de lo contrario} \end{cases} \tag{3.6}$$

donde $|z|$ es la magnitud de z y θ es su fase.

3.5. Arquitecturas base

Los trabajos de Han et al. [11] y Ayat et al. [1] presentan modelos con una estructura similar a CNN pero que operan en la representación frecuencial (Redes de Fourier). Estas arquitecturas reciben como entrada la representación frecuencial de las imágenes, la cual es evaluada por bloques conformados de capas convolucionales, evaluadas por funciones de activación y seguidas de capas de *Spectral Pooling*. Posteriormente, la información obtenida de estos bloques se entrega a capas de neuronas completamente conectadas, para llevar a cabo la clasificación de imágenes. La Figura 3.3 presenta la estructura de las arquitecturas *Efficient Fourier CNN* (EF-CNN) propuesta por Han et al. [11] y *Spectral-Based CNN* (SB-CNN) propuesta por Ayat et al. [1].

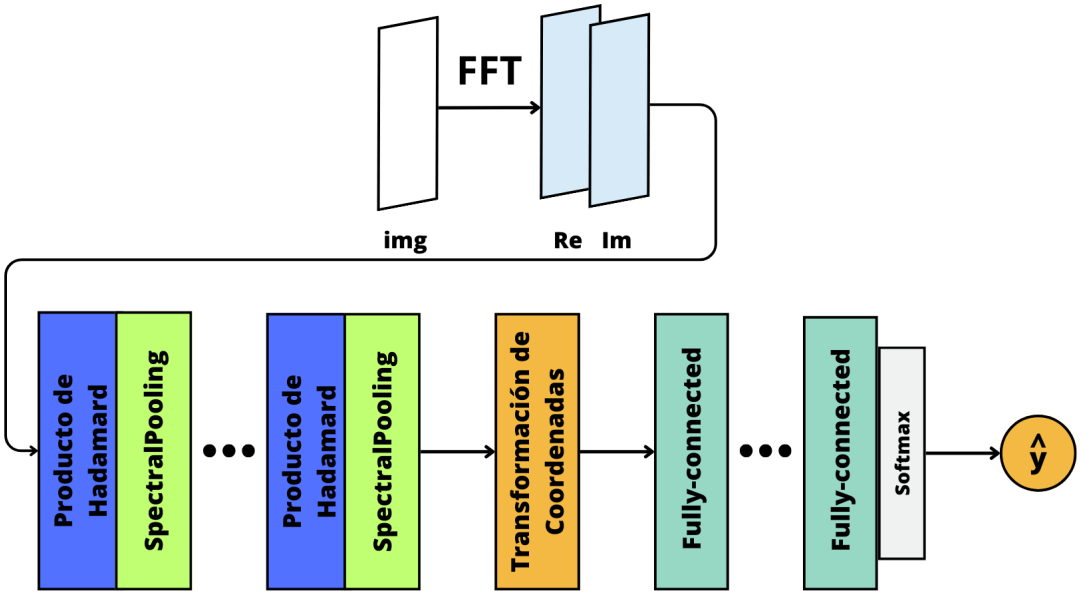


Figura 3.3: Estructura general de las arquitecturas EF-CNN y SB-CNN, conformada por bloques de capas que implementan el producto de Hadamard seguidas de *Spectral Pooling*

Cabe notar que el diseño de ambas arquitecturas tiene el mismo propósito: mantener la estructura del modelo en la representación frecuencial, evitando transformaciones de dominio innecesarias. Sin embargo, este propósito se aborda con técnicas distintas en cada caso. Ambas arquitecturas trabajan con la

representación rectangular de números complejos y utilizan la representación frecuencial no centrada. Sin embargo, las capas de EF-CNN están diseñadas para enfocarse en las esquinas de la representación frecuencial, lo cual es equivalente a trabajar con la representación frecuencial centrada. Por otro lado, las arquitecturas SB-CNN se enfocan en la información contenida en el primer cuadrante de la representación frecuencial (esquina superior izquierda).

En lo que respecta a funciones de activación, EF-CNN utiliza la función PhaseReLU, definida en la Ecuación 3.4, mientras que SB-CNN utiliza la función SReLU, como se define en la Ecuación 3.1. Esta última función involucra a la operación de convolución, por lo cual presenta un elevado costo computacional; por otro lado, PhaseReLU presenta el inconveniente de necesitar una transformación a la representación polar de números complejos. Además, ambas opciones muestran mejoras poco significativas en comparación con otras funciones de activación más simples.

En cuanto a la implementación de la operación de convolución y las capas de *pooling*, ambas arquitecturas aprovechan las propiedades del teorema de la convolución (ver Sección 2.2.2). Dado que ambas arquitecturas reciben la representación frecuencial de las imágenes, la cual es calculada por el algoritmo FFT [14], la convolución se implementa de manera equivalente por medio de capas que calculan el producto de Hadamard entre las entradas y *kernels* de variable compleja. Con el objetivo de reducir el número de parámetros entrenables, Han et al. [11] proponen una técnica basada en *kernels* inicializados aleatoriamente con una distribución Gaussiana, asociados a dos parámetros entrenables que representan la desviación estándar de la distribución. No obstante, este enfoque presenta un gran número de parámetros no entrenables, ya que se requiere almacenar los *kernels* inicializados aleatoriamente. Además, al usar capas de *Spectral Pooling*, las componentes eliminadas por éstas representan parámetros que fueron inicialmente almacenados de forma innecesaria. Por otro lado, las arquitecturas SB-CNN presentan un enfoque en el cual se combina el cálculo del producto de Hadamard con la aplicación de *Spectral Pooling*. En este enfoque, solo se definen los parámetros de las componentes frecuenciales del *kernel* que no serán eliminadas por la capa *Spectral Pooling*. Aunque esta propuesta demuestra una gran reducción de parámetros en comparación con el enfoque de Han et al. [11], aún presenta inconvenientes en las primeras capas de la arquitectura, donde aún se definen parámetros innecesarios.

Antes de pasar a las capas de neuronas completamente conectadas, ambas arquitecturas realizan una transformación para convertir las representaciones bidimensionales en una representación unidimensional. EF-CNN calcula la

magnitud de la reprensión compleja, tal y como se define en la Ecuación 3.7:

$$|z| = \sqrt{\mathcal{R}(z)^2 + \mathcal{I}(z)^2} \tag{3.7}$$

donde $z \in \mathbb{C}$. Por otro lado, SB-CNN calcula la parte real de la transformada de Fourier inversa, definida en la Ecuación 2.7. Finalmente, la clasificación se lleva a cabo utilizando la información obtenida por estas transformaciones usando capas de neuronas completamente conectadas.

En síntesis, se han propuesto diversas propuestas para complementar el uso de arquitecturas CNN, consiguiendo una mejora sustancial en la exactitud. Sin embargo, la tendencia indica que entre mejor sea el desempeño de las arquitecturas mayor será su costo computacional, principalmente debido a la operación de convolución. Para afrontar este inconveniente, surgen diversas propuestas para aligerar el costo computacional de CNN por medio del teorema de la convolución. En estos trabajos destacan técnicas para trabajar en la representación frecuencial, incluyendo funciones de activación en variable compleja y capas de *pooling* para la representación frecuencial. Este enfoque logra reducir el número de operaciones en comparación con CNN, sin embargo, involucra un alto consumo en memoria. Además, las técnicas propuestas para trabajar en la representación frecuencial presentan diversas oportunidades de mejora.

Los modelos propuestos en esta investigación abordan los inconvenientes descritos. Afrontando el inconveniente del alto consumo de memoria y evitando la perdida de información provocada por la capa *Spectral Pooling*.

Capítulo 4

Método Propuesto

En este capítulo se describen dos enfoques distintos: uno para trabajar con imágenes pequeñas, en el cual el tamaño de los *kernels* no implica un gasto elevado de memoria, y otro orientado para trabajar con imágenes grandes, enfocado en la reducción de parámetros en *kernels* de gran tamaño.

4.1. Convolución en la representación frecuencial

En ambos enfoques se reduce el costo de la operación de convolución gracias a las propiedades del teorema de la convolución (Ecuación 2.9), el cual establece que la convolución de un par de señales en su representación espacial es equivalente al producto de Hadamard de las señales en su representación frecuencial.

En primer lugar, es esencial considerar que el producto de Hadamard involucra la multiplicación de componentes frecuenciales, cuyo dominio es complejo. Al trabajar con una representación rectangular compleja de la forma $z = x + iy$, la multiplicación compleja se define como:

$$z_1 \cdot z_2 = (x_1 \cdot x_2 - y_1 \cdot y_2) + i(x_1 \cdot y_2 + x_2 \cdot y_1) \tag{4.1}$$

Por otro lado, si la representación es polar, de la forma $z = r\, e^{i\theta}$, donde r y θ son la magnitud y fase de z respectivamente, la multiplicación compleja se define como sigue:

$$z_1 \cdot z_2 = r_1 \cdot r_2\, e^{i(\theta_1 + \theta_2)} \tag{4.2}$$

En la representación espacial, una capa convolucional se define en función del número de filtros a usar y el número de canales del mapa de características de entrada. Cada *kernel* se define de dimensión arbitraria (usualmente 3×3 o 5×5), pero con el mismo número de canales que la imagen de entrada. De igual manera implementaremos la operación de convolución en la representación frecuencial análogo al caso espacial; cada filtro se define con el mismo número de canales que el mapa de características de entrada. Sin embargo, es importante tener en cuenta que en la representación frecuencial, los *kernels* operarán en un

dominio complejo, el cual involucra dos coordenadas, independientemente de si la representación elegida es rectangular o polar. Además, en la representación frecuencial, la dimensión del *kernel* debe coincidir con la dimensión del mapa de características de entrada. Por esta razón, el número de parámetros entrenables por *kernel* aumenta significativamente en comparación con la representación espacial. Por ejemplo, si se tiene una imagen de entrada de dimensión $A \times A$, por cada filtro espacial cuadrado de dimensión $k \times k$, con $k < A$, se requieren k^2 parámetros entrenables; por otro lado, en el caso de la representación frecuencial son necesarios $2A^2$ parámetros entrenables.

Dado un mapa de características de entrada $I \in \mathbb{C}^{R \times C \times CH}$, donde R, C y CH representan el número de filas, columnas y canales respectivamente, un número de filtros N dado por el usuario y una función de activación compleja no lineal f, el mapa de características de salida Y_j (con $j = 1, 2, \ldots, N$) se calcula como sigue:

$$Y_j = f\left(\sum_{ch=1}^{CH} I_{ch} \odot W_{ch,j}\right) \tag{4.3}$$

donde el índice ch itera sobre la dimensión de los canales, el operador $\odot$ representa el producto de Hadamard complejo y $W_j \in \mathbb{C}^{R \times C \times CH}$ es el *kernel* con el cual opera I. Note que el número de filas, columnas y canales de W, coincide con las dimensiones de I. En este trabajo los experimentos se realizaron con una representación rectangular compleja, por lo cual, el cálculo del mapa de características de salida esta dado por la Ecuación 4.4:

$$Y_j = f\left(\sum_{ch=1}^{CH}\left[\left(I_{ch}^{(re)} \cdot W_{ch,j}^{(re)} - I_{ch}^{(im)} \cdot W_{ch,j}^{(im)}\right) + i\left(I_{ch}^{(re)} \cdot W_{ch,j}^{(im)} + I_{ch}^{(im)} \cdot W_{ch,j}^{(re)}\right)\right]\right) \tag{4.4}$$

donde los superíndices *(im)* y *(re)* representan la parte real e imaginaria de cada componente, respectivamente. En capas posteriores, los N mapas de características de salida representan los canales de un nuevo mapa de características de entrada $I' \in \mathbb{C}^{R \times C \times N}$.

Note que la definición de la Ecuación 4.3 puede expandirse para agregar un termino de *bias* $b_j \in \mathbb{C}$, tanto en la representación rectangular como en la polar:

$$Y_j = f\left(\sum_{ch=1}^{CH} I_{ch} \odot W_{ch,j} + b_j\right) \tag{4.5}$$

4.2. Enfoque para imágenes pequeñas

La propuesta que se plantea se basa en la metodología de las arquitecturas SB-CNN y EF-CNN. El principal inconveniente de estas arquitecturas radica en el

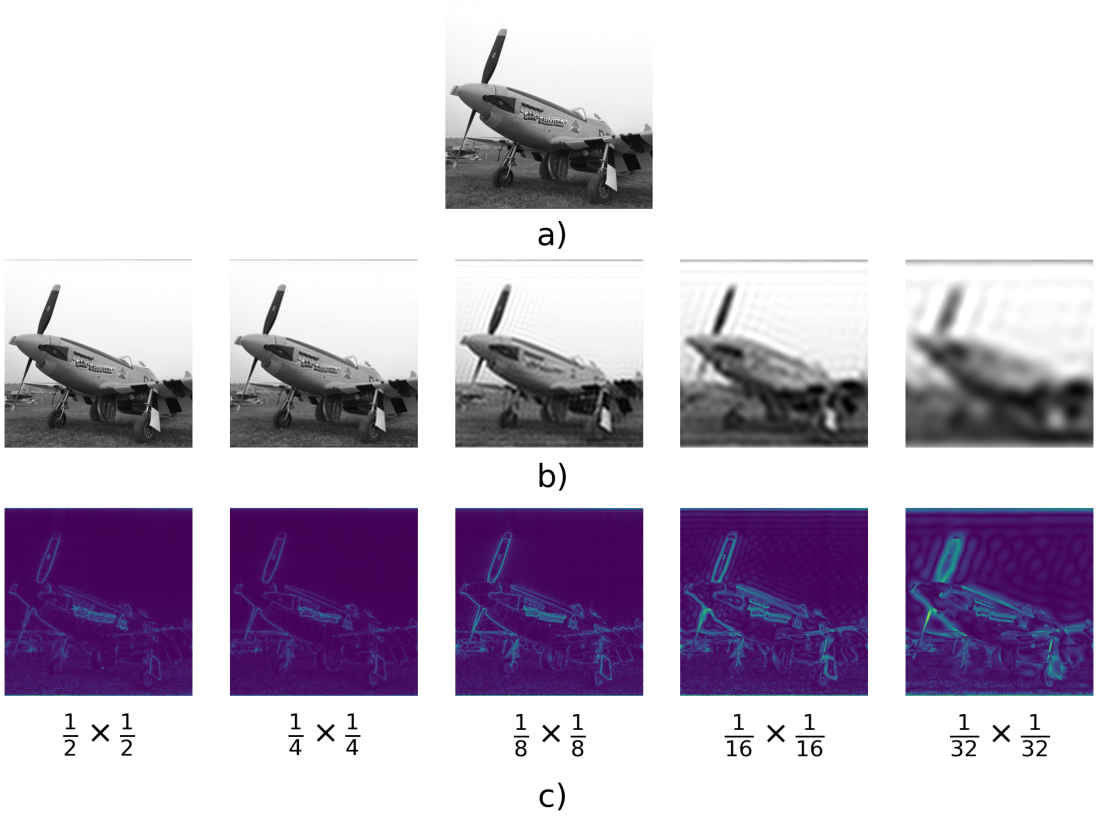


Figura 4.1: Ejemplos del uso de *Spectral Pooling*. a) Imagen original. b) Reconstrucción con la información preservada. c) Reconstrucción con la información descartada

elevado número de parámetros por *kernel*. Ayat et al. [1] y Han et al. [11] sugieren mecanismos para afrontar este inconveniente, sin embargo, el problema seguirá presente mientras se trabaje con representaciones frecuenciales grandes, ya que el número de parámetros por *kernel* en la representación frecuencial depende del tamaño de la representación de entrada.

Las arquitecturas SB-CNN y EF-CNN abordan el problema del elevado número de parámetros al reducir el tamaño de la representación frecuencial mediante la capa de *Spectral Pooling*. Esto conduce a una disminución en los parámetros de las últimas capas convolucionales. Sin embargo, esta capa directamente elimina componentes frecuenciales, resultando en una pérdida de información cada vez que se evalúa la entrada con esta capa. Este inconveniente se hace más evidente en arquitecturas que emplean múltiples capas de *Spectral Pooling*, ya que entre más capas se usen, menos componentes frecuenciales están siendo consideradas.

La Figura 4.1 muestra cómo una imagen es reconstruida al ser procesada por una capa de *Spectral Pooling* con diversos factores de reducción. También se exhibe la reconstrucción de la imagen con la información descartada en cada caso. En los primeros casos, al eliminar pocos componentes frecuenciales, la pérdida

de información es apenas perceptible; sin embargo, al eliminar gran parte de las componentes frecuenciales de la imagen, se observa una gran pérdida de información, principalmente de los bordes de los objetos en la imagen.

Los inconvenientes de las arquitecturas EF-CNN y SB-CNN son abordados mediante la introducción de dos nuevas arquitecturas: tipo-R y tipo-LT. Las primeras reducen el número de parámetros por *kernel* al trabajar con un conjunto reducido de componentes frecuenciales, esto sin disminuir la exactitud de la arquitectura original. Sin embargo, se enfrentan a un dilema de pérdida de información similar al inconveniente observado con *Spectral Pooling*. Por otro lado, las arquitecturas tipo-LT conservan la ventaja de tener un menor número de parámetros y afrontan el problema de pérdida de información al incorporar una nueva capa, la cual tiene un impacto positivo en la exactitud de la arquitectura, pero, involucra un incremento en la cantidad total de parámetros. Este enfoque es apropiado para trabajar con imágenes pequeñas, donde el nivel de detalle no es tan alto. Por consiguiente, un número reducido de componentes serán lo suficientemente representativas para lograr una clasificación precisa.

4.2.1. Arquitecturas tipo-R

Como se mencionó anteriormente, *Spectral Pooling* es la técnica de *pooling* más predominante actualmente para trabajar con imágenes en su representación frecuencial (ver Sección 3.3). A pesar de que *Spectral Pooling* consigue resultados sobresalientes, involucra un desperdicio de recursos debido a la forma en la cual se aplica esta técnica (ver Sección 3.5). En una CNN convencional, una capa convolucional generalmente se acompaña con una capa de *pooling* para reducir el tamaño de la representación. En el caso frecuencial, el concepto sigue siendo similar; sin embargo, *Spectral Pooling* elimina componentes de manera directa, lo que significa que la información de las componentes eliminadas no aporta nada a la arquitectura, como se aprecia en la Figura 4.2.

Para una imagen de entrada en su representación frecuencial, denotada como $\mathtt{Input}$, la convolucion se implementa usando filtros W de variable compleja de la misma dimensión que $\mathtt{Input}$. Por cada filtro convolucional, se realiza el producto de Hadamard entre $\mathtt{Input}$ y W, ya que esta operación es equivalente a la operación de convolución en el dominio espacial. Posteriormente, *Spectral Pooling* elimina las componentes sombreadas en gris, generando una representación de salida $\mathtt{Output}$ de menor dimensión. Las componentes eliminadas por *Spectral Pooling* no contribuyen al aprendizaje de la arquitectura, ya que el cálculo de cada componente del producto de Hadamard es independiente del resto. Por consiguiente, las componentes frecuenciales eliminadas representan un número

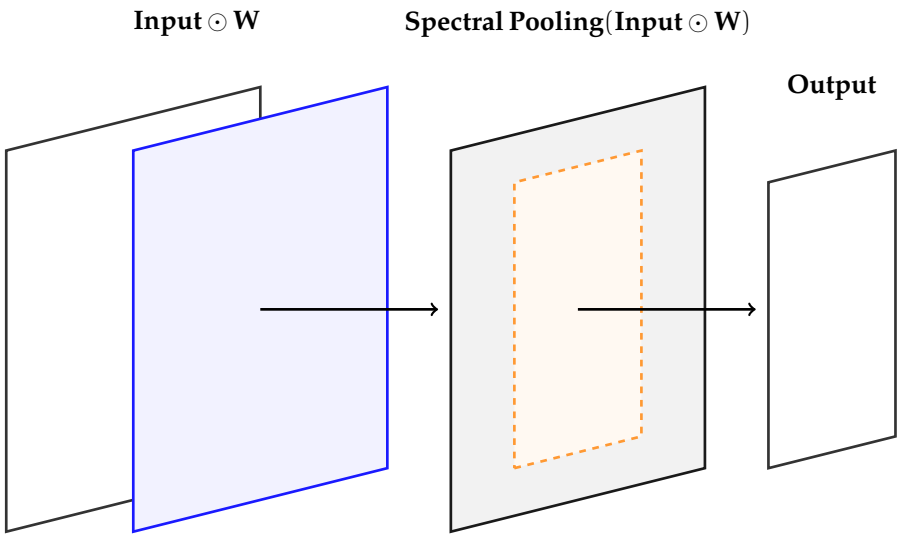


Figura 4.2: Producto de Hadamard seguido de *Spectral Pooling*. La región sombreada representa las componentes eliminadas por *Spectral Pooling*

significativo de parámetros y operaciones de punto flotante innecesarias.

En el caso de arquitecturas profundas como EF-CNN y SB-CNN (ver Sección 3.5), la representación frecuencial de las imágenes se evalúa con bloques de capas que realizan el producto de Hadamard seguido de *Spectral Pooling*. Con cada uno de estos, se produce una reducción del tamaño de la representación frecuencial. Por ejemplo, en el caso de una imagen cuya representación frecuencial es de dimensión $(r_0 \times c_0)$, al evaluarla con m bloques de producto de Hadamard seguidos de capas de *Spectral Pooling*, se producen m reducciones de dimensión, de tal manera que la salida de cada bloque es de dimensión $(r_1 \times c_1)$, $(r_2 \times c_2)$, ... , $(r_m \times c_m)$, respectivamente. Dado que el cálculo de cada una de las componentes del producto de Hadamard es independiente del resto, las únicas componentes que realmente aportan a la arquitectura son aquellas contenidas en la región $r_m \times c_m$, como se aprecia en la Figura 4.3a, donde las componentes sombreadas representan parámetros entrenables y operaciones innecesarias.

Para afrontar este inconveniente, proponemos utilizar únicamente la región de componentes frecuenciales que genuinamente contribuyen al aprendizaje de la arquitectura, la cual denominaremos representación reducida. Considerando el ejemplo anterior, esta técnica consiste en usar únicamente la región de componentes frecuenciales de dimensión $r_m \times c_m$, resultantes de la última capa de *Spectral Pooling*, tal y como se ilustra en la Figura 4.3b.

Para implementar una arquitectura tipo-R, tomando como base cualquier Red de Fourier, es esencial considerar si se está utilizando una representación frecuencial centrada o no, para construir la representación reducida con la información más adecuada. Por ejemplo, en el caso de las Redes de Fourier

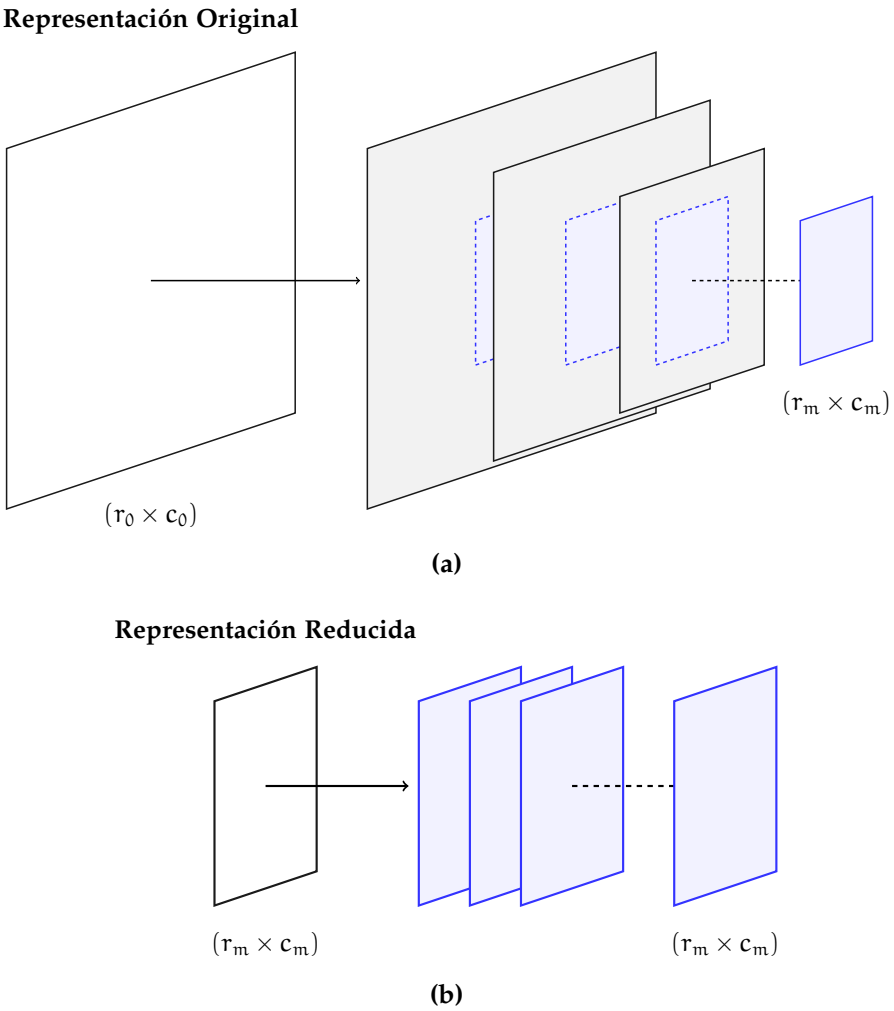


Figura 4.3: a)Reducción de dimensión Redes de Fourier usando capas de *Spectral Pooling*, con las componentes eliminadas representadas por la región sombreada. b) Método propuesto eliminando capas de *Spectral Pooling* y usando únicamente las componentes que aportan a la arquitectura

EF-CNN y SB-CNN, que se replicaron usando una representación frecuencial centrada y una representación frecuencial normal, respectivamente. En el primer caso, la representación reducida se forma con la región de componentes cubiertas por una ventana centrada de dimensión $M \times N$. Para una representación frecuencial no centrada, la representación reducida se construye de la misma manera pero con la ventana ubicada en la esquina superior izquierda. En ambos casos, la dimensión de salida de la última capa de *Spectral Pooling* en la arquitectura original determina el valor de los parámetros M y N. La arquitectura tipo-R recibirá como entrada esta representación reducida en lugar de la representación frecuencial completa, como se ilustra en la Figura 4.4. El resto de la arquitectura tipo-R mantiene el mismo número de capas convolucionales y su configuración, incluyendo funciones de activación y número de filtros,

pero omite el uso de capas de *Spectral Pooling*. El diseño de las arquitecturas tipo-R garantiza que el aprendizaje se realice con la misma información que la arquitectura base, lo cual asegura la misma exactitud en ambos casos. Esta afirmación se respalda experimentalmente en la Sección 5.1. Cabe mencionar que, en la práctica, se tiene la flexibilidad de seleccionar cualquier dimensión para la representación reducida, e incluso, optar por diferentes componentes frecuenciales según lo que resulte más conveniente.

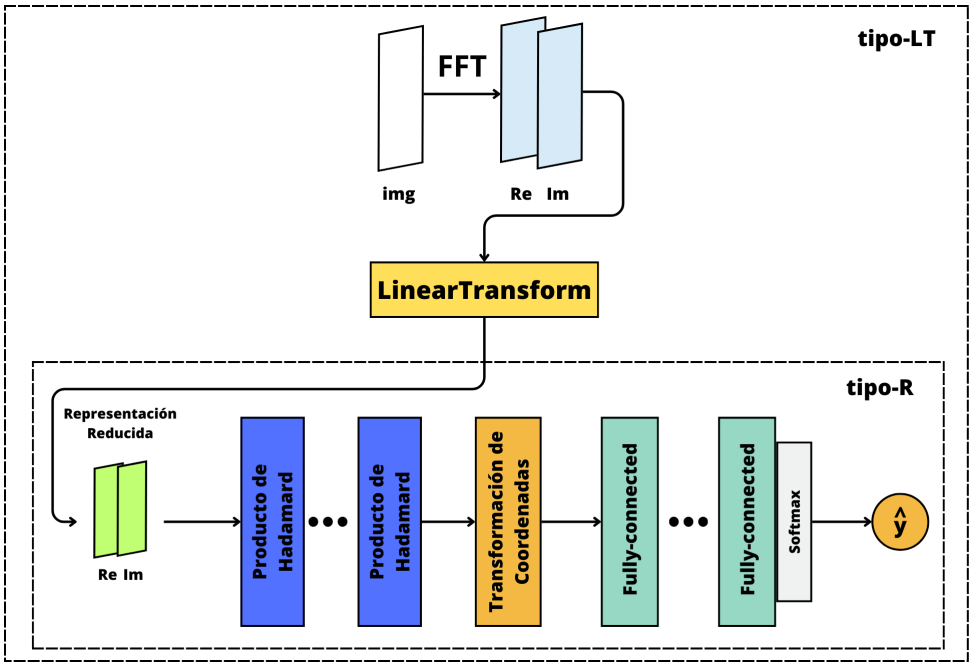


Figura 4.4: Estructura de las arquitecturas tipo-R y tipo-LT. Mientras que la primera opción recibe una representación reducida, las arquitecturas tipo-LT construyen esta representación con la capa *Linear Transform*

La equivalencia entre las arquitecturas de Redes de Fourier base y sus equivalentes tipo-R sugiere que un conjunto reducido de componentes frecuenciales contienen información suficiente para realizar una clasificación precisa. Sin embargo, en la práctica, determinar cuáles son los componentes frecuenciales más relevantes en cada conjunto de datos puede resultar complicado. Los planteamientos de las arquitecturas EF-CNN y SB-CNN sugieren que los componentes más relevantes de cualquier conjunto de datos suelen ubicarse en la región central de la representación frecuencial centrada o en el primer cuadrante de la representación frecuencial normal. Sin embargo, no se descarta la posibilidad de que existan componentes frecuenciales más significativos en regiones distintas, y que su ubicación varíe en cada conjunto de datos.

Existe un compromiso entre el costo computacional de las arquitecturas tipo-R y su exactitud. Disminuir el tamaño de la representación reducida resulta en un ahorro en parámetros y operaciones, pero también disminuye la capacidad de

los *kernels* para identificar patrones complicados, lo que repercute en una menor exactitud. Por otro lado, se espera que el uso de una representación reducida de mayor tamaño permita a los *kernels* identificar características de alta complejidad gracias a su nivel de detalle, potencialmente mejorando el rendimiento de la arquitectura. Sin embargo, esto conlleva un aumento en el número de parámetros y operaciones de punto flotante.

4.2.2. Arquitecturas tipo-LT

Como se mencionó previamente, la efectividad de las arquitecturas tipo-R sugiere que basta con un conjunto reducido de componentes frecuenciales para obtener buenos resultados. Sin embargo, en la práctica es complicado determinar cuáles son las componentes frecuenciales más representativas para construir una representación reducida en cada conjunto de datos. Por otro lado, al usar una representación reducida pequeña, existe el riesgo de perder información valiosa contenida en componentes frecuenciales no consideradas, lo que podría resultar en una clasificación menos precisa. Por estas razones se proponen las arquitecturas tipo-LT, en conjunto con una capa que aprende durante el proceso de entrenamiento a identificar las componentes frecuenciales más adecuadas para construir una representación reducida optimizada, considerando toda la información de la representación frecuencial de la imagen de entrada.

La nueva capa propuesta, denominada *Linear Transform*, aprovecha toda la información de la representación frecuencial transformándola linealmente en una representación reducida optimizada, como se ilustra en la Figura 4.5. Por ejemplo, en el caso de una representación de entrada de dimensión $A \times A$ con la cual se desea construir una representación reducida de dimensión $B \times B$, siendo $B < A$, se divide la representación frecuencial de entrada en cuadrantes, de dimensión $\frac{A}{2} \times \frac{A}{2}$. Cada cuadrante se redimensiona en una representación unidimensional de tamaño $1 \times \left(\frac{A}{2}\right)^2$ y se transforma mediante una operación de multiplicación matricial con una matriz de parámetros entrenables de tamaño $\left(\frac{A}{2}\right)^2 \times \left(\frac{B}{2}\right)^2$, denominada matriz de transformación. La representación resultante, de dimensión $1 \times \left(\frac{B}{2}\right)^2$, se transforma en una representación de dimensión, $\frac{B}{2} \times \frac{B}{2}$. Finalmente, el resultado de la transformación de los cuatro cuadrantes se combina en una representación de dimensión $B \times B$. Las matrices de transformación se construye con parámetros entrenables, por lo cual son capaces de adaptarse durante el entrenamiento para conseguir una configuración que favorezca a la clasificación. En este ejemplo, la capa *Linear Transform* requiere de $\frac{A^2 \times B^2}{4}$ parámetros entrenables para caracterizar las cuatro matrices de transformación necesarias. Cabe mencionar que la multiplicación matricial afecta de igual manera

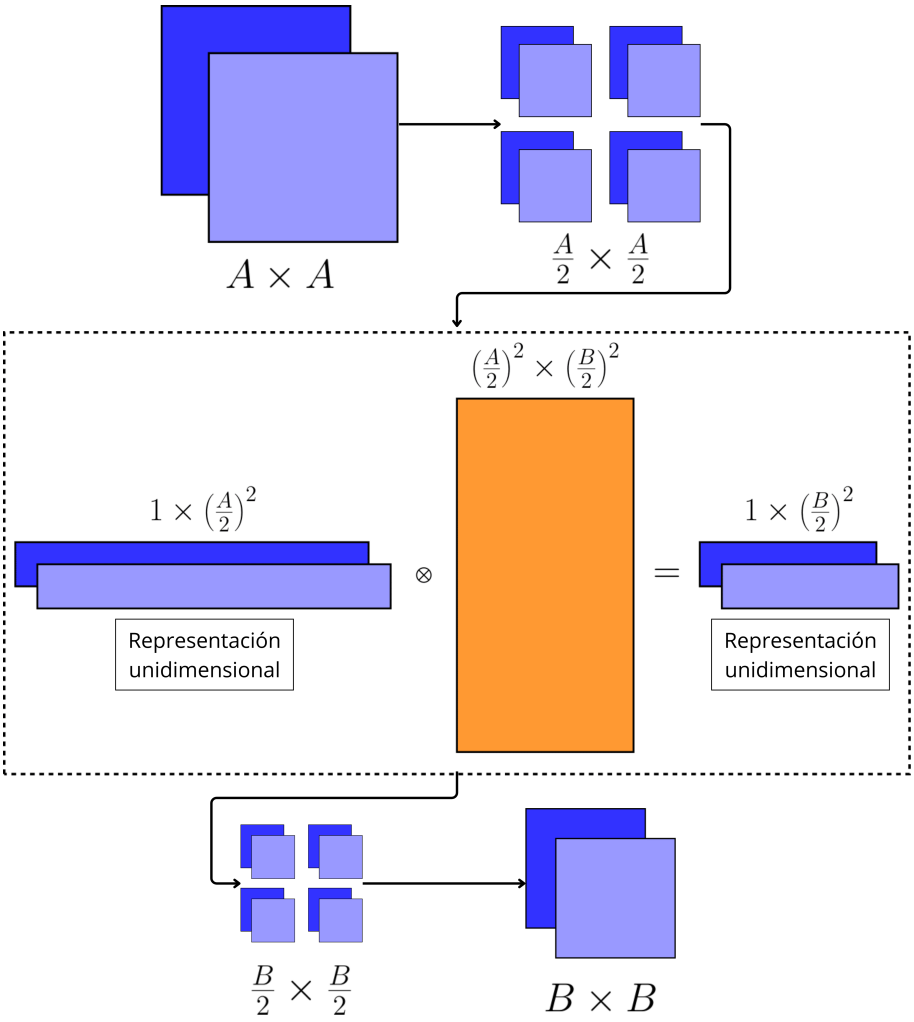


Figura 4.5: Mecanismo de la capa *Linear Transform* aplicado a cada cuadrante en la representación frecuencial, con el fin de reducir su tamaño considerando toda la información disponible

a la parte real e imaginaria de la representación frecuencial de entrada. Además, esta operación es una transformación lineal, por lo cual, se espera que la representación resultante no pierda las propiedades del dominio de Fourier, incluyendo las cualidades del teorema de la convolución.

La capa *Linear Transform* desempeña un papel central en las arquitecturas tipo-LT propuestas. Estas arquitecturas poseen una estructura similar a las arquitecturas tipo-R, como se ilustra en la imagen 4.4. Sin embargo, la representación reducida con la que trabaja esta arquitectura es construida por una capa *Linear Transform*. Las arquitecturas tipo-LT están diseñadas para aprovechar toda la información de la representación frecuencial original, afrontando el inconveniente de pérdida de información de componentes no consideradas en una representación reducida. Además, resultados experimentales muestran que la capa *Linear Transform* es capaz de extraer información igualmente representativa

que representaciones reducidas de mayor tamaño, lo cual impacta positivamente en la exactitud de la arquitectura. Por lo tanto, las capas *Linear Transform* son idóneas para trabajar con representaciones reducidas pequeñas sin comprometer la información, lo que permite reducir el número de parámetros entrenables por *kernel*, ya que entre menor sea el tamaño de la representación, menor será el número de parámetros necesarios.

4.3. Enfoque para imagen grandes

Algunos conjuntos de datos contienen imágenes de alta resolución, ya que se requiere un alto nivel de detalle para identificar las características necesarias para realizar la clasificación de estas imágenes. Este es el caso, por ejemplo, de las imágenes naturales o médicas.

Las arquitecturas tipo-R y tipo-LT presentan problemas de memoria al trabajar con este tipo de imágenes. Esto se debe a que, a medida que aumenta el nivel de detalle de las imágenes, también lo hace el número de componentes frecuenciales que deben considerarse para crear una representación reducida lo suficientemente representativa para realizar la clasificación. Por consiguiente, el tamaño de los *kernels* tambien crece, lo cual implica un mayor consumo de memoria RAM para trabajar con estas arquitecturas. En la práctica se observan constantes interrupciones en la ejecución durante el entrenamiento de estas arquitecturas por problemas de memoria RAM, atribuidos a la gran cantidad de parámetros entrenables que deben almacenarse y a la memoria consumida por el algoritmo de optimización al calcular gradientes.

Para afrontar estos inconvenientes, se propone otro enfoque con características similares a las CNN. En este enfoque se aplica la operación de convolución utilizando la misma estrategia descrita en la Sección 4.1. Sin embargo, se plantea una estrategia para generar *kernels* con pocos parámetros entrenables, independientemente del tamaño de la representación frecuencial, similar a los *kernels* de CNN, donde el número de parámetros no depende del tamaño de la imagen de entrada. Esta técnica se basa en los filtros Butterworth comunmente utilizados en el área de DSP. Además, para reducir el tamaño de la representación frecuencial, se presenta una nueva capa de *pooling*, denominada *Spectral Average Pooling*, la cual, a diferencia de *Spectral Pooling*, no elimina componentes de manera directa, lo cual evita pérdida de información.

Estas nuevas técnicas se incorporan en las arquitecturas Butterworth CNN (BW-CNN), las cuales siguen una estructura similar a CNN, compuestas de bloques de capas convolucionales con filtros Butterworth y capas *Spectral Average*

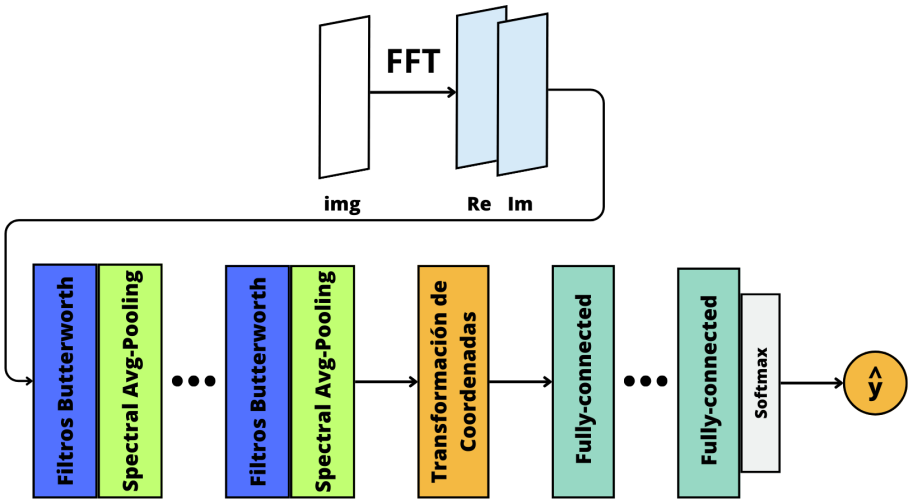


Figura 4.6: Estructura de las arquitecturas BW-CNN, compuestas por bloques de capas que implementan la técnica de filtros Butterworth, seguidos de *Spectral Average Pooling*

Pooling, como se ilustra en la Figura 4.6.

4.3.1. Filtros Butterworth

La técnica que se propone, denominada filtros Butterworth replica el proceso de filtrado de imágenes en el dominio de Fourier, una técnica ampliamente utilizada en el área de procesamiento de señales (ver Sección 2.3). Este enfoque se especializa en la generación de filtros para trabajar con la representación frecuencial, donde su diseño resulta más intuitivo.

En este estudio, nos centraremos específicamente en los filtros Butterworth, los cuales, a diferencia de lo filtros ideales pasa-bajas/altas descritos en la Sección 2.3, se caracterizan por ser parametrizados mediante una función continua y derivable, lo cual es ideal para implementar esta técnica en una red neuronal. Además, en contraste con los filtros ideales que consisten únicamente en ceros y unos, los filtros Butterworth tienen valores reales en el conjunto $[0, 1]$. Por lo tanto, estos filtros ofrecen una mayor flexibilidad para asignar la intensidad más apropiada a cada componente frecuencial.

Para parametrizar estos filtros, se propone una función generadora de filtros, basada en la definición convencional de filtros Butterworth [24]. La modificación propuesta, $G(x, y)$, se define en la Ecuación 4.6:

$$G(m, n) = \frac{A}{1 + \frac{1}{g_0^2}\left[(m - m_0)^2 + (n - n_0)^2\right]^d} + b \tag{4.6}$$

donde m_0 y n_0 son las coordenadas del centro del filtro, d es el grado de la función (ver Figura 4.7), g_0 parametriza el punto de inflexión de la curva y las

constantes A y b sirven como variables auxiliares para determinar la curvatura del filtro. Si $A = 1$ y $b = 0$, entonces el filtro tendrá una curvatura cóncava, por otro lado, si $A = -1$ y $b = 1$, entonces la curvatura del filtro será convexa.

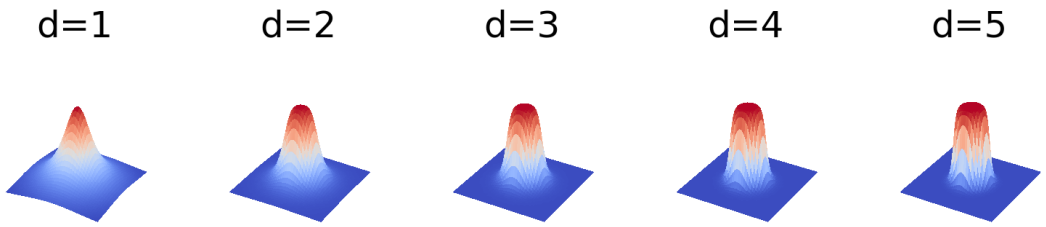


Figura 4.7: Filtros Butterworth con distintos grados de curvatura (d)

Dependiendo de la curvatura de G y de si la posición de m_0 y n_0 está en el centro, se generan distintos tipos de filtros. Estas combinaciones se describen la Tabla 4.1 y la Figura 4.8 ilustra el resultado de evaluar una imagen con cada tipo de filtro.

Tabla 4.1: Tipos de filtros generados por la función generadora de filtros Butterworth $G(m, n)$

A	b	(m_0, n_0)	Filtro
1	0	$m_0 = 0, n_0 = 0$	pasa-bajas
-1	1	$m_0 = 0, n_0 = 0$	pasa-altas
1	0	$m_0 \neq 0, n_0 \neq 0$	pasa-bandas
-1	1	$m_0 \neq 0, n_0 \neq 0$	rechaza-bandas

Dada una representación frecuencial de entrada $F \in \mathbb{C}^{r \times c}$, la función generadora $G(m, n)$ dinámicamente crea un filtro Butterworth al evaluar una cuadricula de valores Δ con la función $G(m, n)$. Δ se define en la Ecuación 4.7:

$$\begin{aligned}\Delta = &[-dim, -dim + dr, -dim + 2 \cdot dr, \ldots, dim - 2 \cdot dr, dim - dr, dim] \\ &\times [-dim, -dim + dc, -dim + 2 \cdot dc, \ldots, dim - 2 \cdot dc, dim - dc, dim] \end{aligned} \quad (4.7)$$

donde $dr = \frac{2 \cdot dim}{r}$, $dc = \frac{2 \cdot dim}{c}$ y dim representa la escala de valores considerados para la cuadricula. Note que el número de componentes en la cuadricula coincide con la dimensión de F, por lo cual el nivel de detalle del filtro depende de la dimensión de la representación frecuencial de entrada F. Esta técnica se implementa en una capa denominada BW-Layer, que a su vez realiza la operación de convolución en la representación frecuencial como se describió en la Sección 4.1. Al instanciar una capa BW-Layer, se tiene el control sobre varios aspectos,

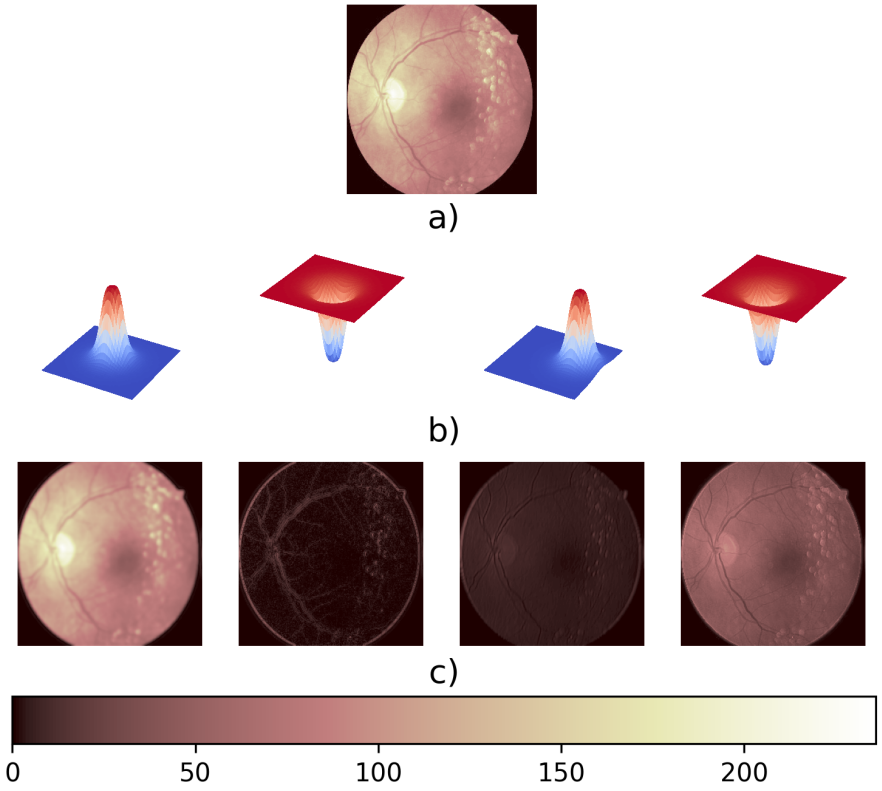


Figura 4.8: Resultado de convolucionar una imagen con distintos tipos de filtros Butterworth. a) Imagen original. b) Filtros pasa-bajas, pasa-altas, pasa-bandas y rechaza-bandas, respectivamente. c) Resultado de convolucionar la imagen original con el filtro ubicado en la parte superior

incluyendo el grado de la función generadora (n), la dimensión de la cuadrícula que parametrizará a la curva (dim) y la distancia del centro del filtro al punto de inflexión (g_0). Para cada filtro en la capa BW-Layer, se decide aleatoriamente la curvatura del filtro con las combinaciones de valores para A y b, donde $A = 1$ y $b = 0$, o bien, $A = -1$ y $b = 1$; estos parámetros permanecen como no entrenables. Por otro lado, a cada filtro se le asignan dos parámetros entrenables, m_0 y n_0, inicializados aleatoriamente. Para ello, se generan coordenadas aleatorias dentro del área cubierta por un circulo de radio R, con R dado por el usuario. Los parámetros m_0 y n_0 permiten ajustar la posición del filtro para identificar las características más sobresalientes para realizar la clasificación.

Cabe notar que en total esta capa posee únicamente 4 parámetros por filtro, independientemente del tamaño de la representación frecuencial de entrada. En contraste con las arquitecturas tipo-R y tipo-LT, donde trabajar con imágenes grandes involucraría un elevado número de parámetros. Además, esta técnica mantiene un número de parámetros comparable o menor que una arquitectura CNN, dependiendo del tamaño de los *kernels* usados.

4.3.2. *Spectral Average Pooling*

Para complementar el uso de filtros Butterworth proponemos una nueva capa de *pooling* para la representación frecuencial. El propósito de esta capa es disminuir el tamaño de la representación, al igual que cualquier capa de *pooling*, pero sin perder información al eliminar componentes de manera directa. Este enfoque contrasta con la capa *Spectral Pooling*, cuyo mecanismo involucra una alta pérdida de información, ya que entre más capas se utilicen menos componentes frecuenciales están siendo consideradas en el proceso de aprendizaje de la red.

La nueva capa propuesta, denominada *Spectral Average Pooling*, implementa un mecanismo similar al de la capa *Average Pooling* convencional, pero en este caso operando en variable compleja. Como se reviso en la Sección 2.1.2, la capa *Average Pooling* divide una imagen de entrada en ventanas y calcula su promedio. Del mismo modo, buscamos replicar este mecanismo en la representación frecuencial.

Se define el promedio de un conjunto de n variables $\{v_1, v_2, \ldots, v_n\}$ como la suma de todas las variables dividida entre el número elementos en el conjunto:

$$Avg\left(\{v_1, v_2, \ldots, v_n\}\right) = \frac{1}{n}\sum_{i=1}^{n} v_i \tag{4.8}$$

Análogamente, se define el promedio de un conjunto de n números complejos $\{c_1, c_2, \ldots, c_n\}$, con $c_j = x_j + i\, y_j$, como sigue:

$$\begin{aligned} CAvg\left(\{c_1, c_2, \ldots, c_n\}\right) &= \frac{1}{n}\sum_{j=1}^{n} c_j \\ &= \frac{1}{n}\sum_{j=1}^{n} (x_j + i\, y_j) \\ &= \frac{1}{n}\sum_{j=1}^{n} x_j + i\frac{1}{n}\sum_{j=1}^{n} y_j \\ &= Avg\left(\{x_1, x_2, \ldots x_n\}\right) + i\, Avg\left(\{y_1, y_2, \ldots y_n\}\right) \end{aligned} \tag{4.9}$$

Note que el cálculo del promedio de números complejos puede realizarse de forma independiente para la parte real e imaginaria. Por lo cual, se deduce que el mecanismo de la capa *Average Pooling* puede replicarse al aplicar esta capa por separado a las partes real e imaginaria de la representación compleja de entrada, como se ilustra en la Figura. 4.9.

Dada una representación compleja de entrada $F \in \mathbb{C}^{A\times A}$, la capa *Spectral Average Pooling* separa la información en su parte real e imaginaria. Posteriormente aplica una capa de *Average Pooling* con *stride* 2×2 a cada componente, resultando en una representación compleja de dimensión $\frac{A}{2} \times \frac{A}{2}$. La configuración del *stride*

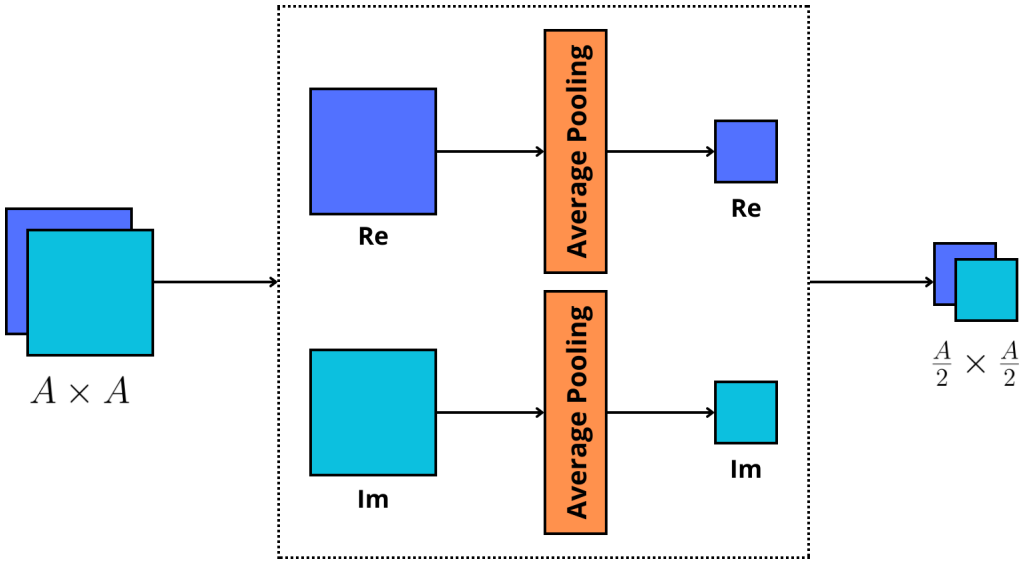


Figura 4.9: Mecanismo de *Spectral Average Pooling*, aplicando la transformación *Average Pooling* por separado a la parte real e imaginaria

de las capas de *Average Pooling* está diseñada para que el cálculo de los promedios se realice entre regiones independientes de dimensión 2×2. Esto se hace con el propósito de no mezclar la información de regiones diferentes, ya que cada componente de la representación frecuencial posee información muy específica.

Es importante destacar que, en la práctica, la capa *Spectral Average Pooling* pierde la estructura de la representación espacial, como se aprecia en la Figura 4.10. No obstante, experimentalmente se observa su efectividad el trabajar en la representación frecuencial.

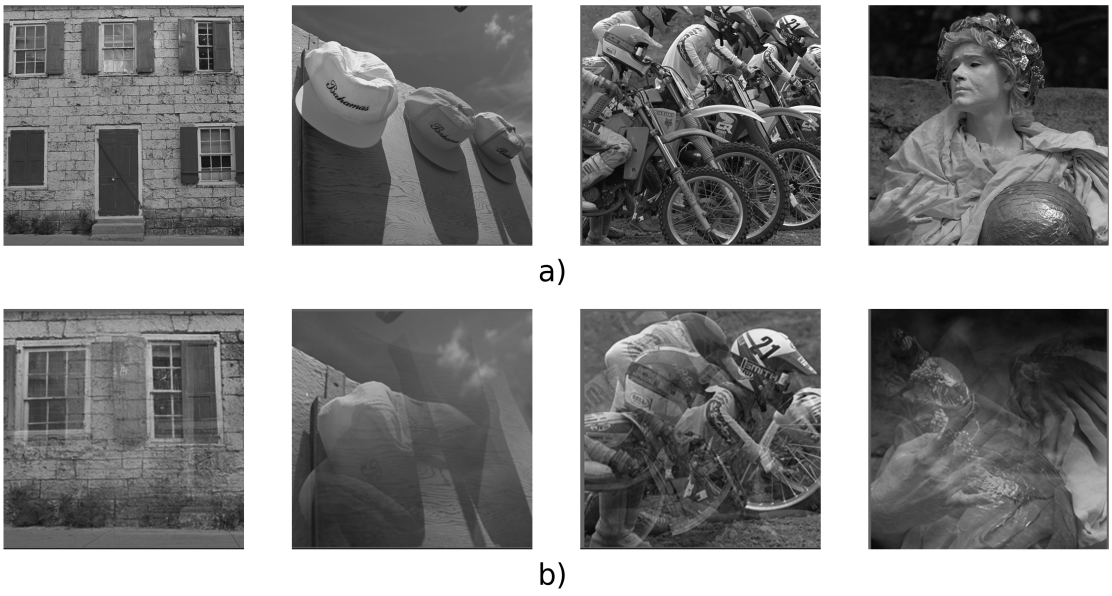


Figura 4.10: Ejemplos del uso de *Spectral Average Pooling*. a) Imágenes originales. b) Resultado al usar *Spectral Average Pooling*

En resumen, en esta investigación se proponen dos enfoques, uno para imágenes en baja resolución y otro para alta resolución. En el primer enfoque se proponen las arquitecturas tipo-R y tipo-LT. La primera opción trabajan con una representación reducida de las componentes frecuenciales de una imagen y elimina el uso de capas *Spectral Pooling*. Análogamente, las arquitecturas tipo-LT trabajan de manera similar, pero crean dinámicamente una representación reducida optimizada con ayuda de la capa *Linear Transform*

En el enfoque para imágenes de alta resolución, se propone un enfoque basado en una función generadora de filtros Butterwoth, que caracteriza kernels con 4 parámetros, independientemente del tamaño de la imagen de entrada. Además, se propone la capa de *Spectral Average Pooling* para complementar el uso de esta técnica.

Capítulo 5

Experimentos y resultados

En esta sección se muestran los experimentos realizados para evaluar las arquitecturas propuestas. Los experimentos se dividen en dos secciones, una enfocada en imágenes pequeñas (Sección 4.2) y otra en imágenes grandes (Sección 4.3). Esta división se realiza para estudiar por separado cada arquitectura propuesta en el escenario para el cual fueron diseñadas. En el primer caso, se evalúan las arquitecturas tipo-R y LT. Por otro lado, en el enfoque para imágenes grandes, se evalúan las arquitecturas BW-CNN. Para ambos enfoques, se evalúa la exactitud obtenida (ver Ecuación 2.19), el número de parámetros en cada arquitectura y el número operaciones de punto flotante (FLOP, por sus siglas del inglés) realizadas durante la inferencia. Se comparan las arquitecturas propuestas con algunas Redes de Fourier propuestas en el estado del arte y con arquitecturas CNN convencionales, todas descritas en el Capítulo 2.

Las Redes de Fourier y las arquitecturas propuestas se implementaron usando capas personalizadas de Keras, con la versión 2.10 de Tensorflow; todos los experimentos se ejecutaron en un equipo con las siguientes especificaciones: una tarjeta gráfica RTX 3060 de 12GB de VRAM, un procesador Ryzen 5 3600 y 16 GB de RAM. Es importante hacer notar que algunos resultados similares a los presentados en este capítulo forman parte de un artículo sometido a la revista *Neural Computing and Applications* el cual, al momento de publicarse este documento, se encontraba en revisión.

5.1. Enfoque para imágenes pequeñas

En esta sección se presentan los resultados obtenidos al evaluar las arquitecturas tipo-R y LT, descritas en la Sección 4.2. Estas técnicas se aplican tomando como base a las arquitecturas EF-CNN y SB-CNN, propuestas por Han et al. [11] y Ayat et al. [1], respectivamente, y descritas en la Sección 3.5. Para todos los experimentos, se evalúa la exactitud y el ahorro en el número de parámetros y operaciones de punto flotante, en comparación con las arquitecturas base.

5.1.1. Diseño experimental

Para los experimentos presentados en esta sección se utilizaron los conjuntos de datos MNIST [33], Fashion-MNIST [34] y CIFAR-10 [35]. Tanto MNIST como Fashion-MNIST constan de $70,000$ instancias de tamaño 28×28, divididas en $60,000$ instancias para entrenamiento (*training*) y $10,000$ para prueba (*testing*). CIFAR-10 contiene $60,000$ instancias de tamaño 32×32, de las cuales $50,000$ se destinan al entrenamiento y $10,000$ para prueba. Estos conjuntos de datos poseen 10 clases balanceadas. MNIST y Fashion-MNIST contienen imágenes en escala de grises, por lo cual las arquitecturas en el dominio de Fourier se evalúan con la transformada rápida de Fourier bidimensional de estas imágenes. En el caso de CIFAR-10, las imágenes se encuentran en formato RGB, por lo que, con el fin de realizar comparaciones justas con las arquitecturas base, se analiza este conjunto tanto en escala de grises (representado en las tablas como CIFAR-10G), como en formato RGB (representado en las tablas como CIFAR-10RGB). En este último caso, se calcula la transformada rápida de Fourier bidimensional por separado para cada canal de la imagen.

Las arquitecturas tipo-R que utilizan como base a las arquitecturas EF-CNN se denominan R-EF-CNN, mientras que las varaintes tipo-LT se llaman LT-EF-CNN. En el caso de las arquitecturas SB-CNN, se nombran R-SB-CNN para el caso de modificaciones usando una arquitectura tipo-R y LT-SB-CNN para las variantes tipo-LT. A continuación se describen detalladamente cada una de las arquitecturas involucradas:

- CNN: Esta arquitectura recibe como entrada las imágenes en representación espacial. La arquitectura contiene 3 capas convolucionales de 6, 16 y 120 filtros de tamaño 5×5, respectivamente. Cada capa convolucional se evalúa usando ReLU y se acompaña de una capa de *Max Pooling*. Después de la tercera capa de *Max Pooling*, sigue una capa *Flatten* que conecta con una capa de *Batch Normalization*. A la salida de esta capa se conectan dos capas de neuronas completamente conectadas: la primera de 84 neuronas con función de activación ReLU, y la última de 10 neuronas con función de activación *softmax*, utilizada para realizar la clasificación.

- EF-CNN y SB-CNN: ambas arquitecturas reciben como entrada la representación frecuencial centrada de las imágenes (ver Sección 2.2.3). Aunque SB-CNN originalmente se enfoca en las componentes frecuenciales del primer cuadrante, se han observado mejores resultados experimentalmente al emplear las componentes frecuenciales centrales, por lo que esta configuración es utilizada aquí. Ambas arquitecturas

poseen la misma configuración de capas convolucionales que CNN, pero la convolución se realiza como se describe en la Sección 4.1. Para EF-CNN, se implementa la técnica de *kernels* aleatorios descrita en [11], mientras que para SB-CNN, los *kernels* se implementan con la técnica descrita en [1], a fin de reducir el número de parámetros. Cada capa convolucional se evalúa con CReLU (ver Ecuación 3.3) y se acompañan de una capa de *Spectral Pooling*. Tras la tercera capa de *Spectral Pooling*, ambas arquitecturas realizan una transformación de coordenadas. EF-CNN calcula la magnitud al cuadrado y SB-CNN la transformada de Fourier inversa. El resto de la configuración de ambas arquitecturas es idéntica a la de CNN.

- R-EF-CNN y R-SB-CNN: dado que las arquitecturas base CNN, EF-CNN y SB-CNN producen una representación de tamaño 4×4 después de la tercera capa de *pooling*, las variantes R-EF-CNN y R-SB-CNN reciben como entrada una representación reducida de tamaño 4×4, formada por la región de componentes centrales de la representación frecuencial centrada. Estas variantes mantienen la misma configuración que sus arquitecturas base EF-CNN y SB-CNN, con la distinción de que se eliminan las capas de *Spectral Pooling*.

- LT-EF-CNN y LT-SB-CNN: estas variantes reciben como entrada la representación frecuencial completa de las imágenes y la convierten en una representación reducida de tamaño 4×4, utilizando una capa *Linear Transform*. El resto de las arquitecturas poseen la misma configuración que las variantes tipo-R.

Es importante destacar que el tamaño de la representación antes de la capa *Flatten* es idéntico en todas las arquitecturas. Por lo tanto, las capas de neuronas completamente conectadas reciben una representación de igual tamaño, asegurando así que la clasificación se realice con la misma cantidad de información, por consiguiente, se garantiza una comparación justa entre estas arquitecturas.

En cuanto al proceso de entrenamiento, todas los experimentos en esta sección se realizaron entrenando durante 50 épocas, con un optimizador *Stochastic Gradient Descent* (SGD) con coeficiente de aprendizaje de 0,0005, *Momentum* de 0,9 y tamaño de *batch* 128.

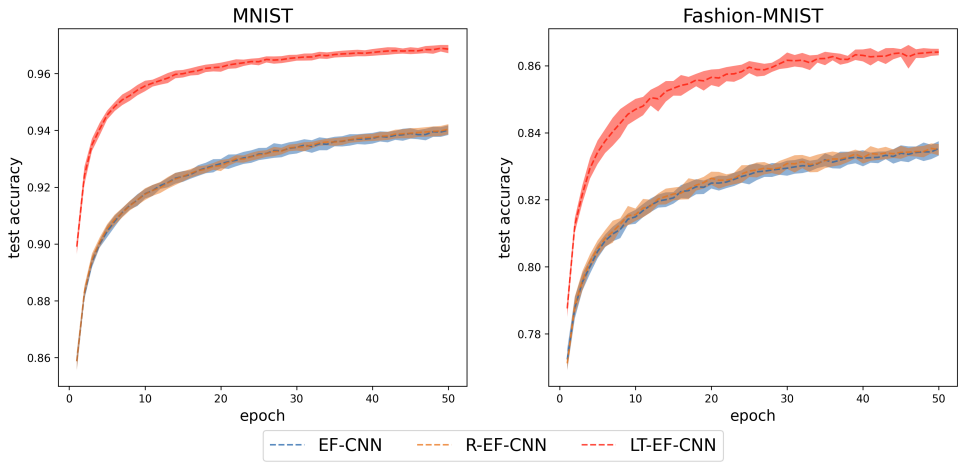
5.1.2. Evaluación de exactitud

Se realizaron dos experimentos distintos para evaluar el desempeño de las arquitecturas propuestas. En el primero, se compara la exactitud de las

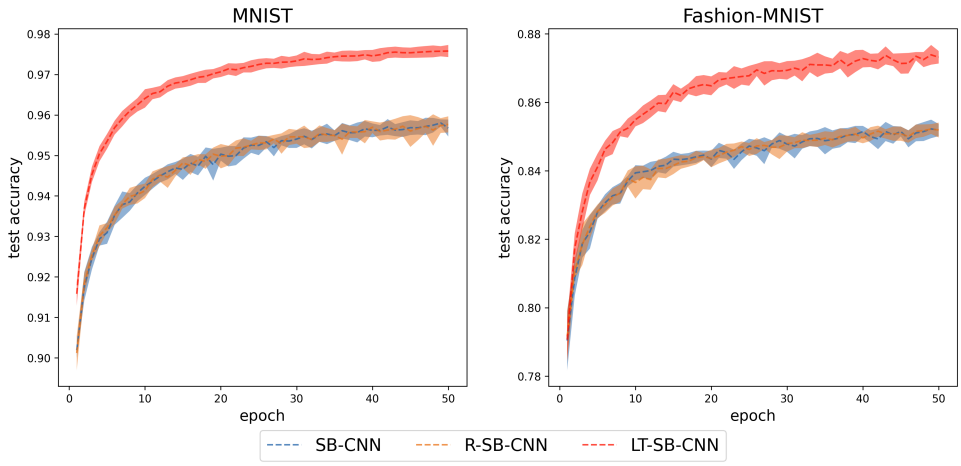
arquitecturas base EF-CNN y SB-CNN con sus variantes tipo-R y tipo-LT. En el segundo experimento se compara la exactitud de las Redes de Fourier en comparación con CNN; además se examina si existe una mejora en la exactitud de las arquitecturas tipo-R y LT, al trabajar con representaciones reducidas de mayor tamaño.

Primer experimento

Aquí se empleó la división por defecto de datos entrenamiento-prueba proporcionada en cada conjunto (ver Sección 5.1.1). Se realizaron 10 experimentos con las arquitecturas e hiperparámetros descritos en la Sección 5.1.1. La Figura 5.1 muestra los resultados experimentales de MNIST y Fashion-MNIST, donde se gráfica la exactitud promedio obtenida al evaluar el conjunto de prueba en cada época, con la región sombreada indicando la desviación estándar.



(a) Arquitecturas EF-CNN, R-EF-CNN y LT-EF-CNN



(b) Arquitecturas SB-CNN, R-SB-CNN y LT-SB-CNN

Figura 5.1: Exactitud promedio en cada época en el conjunto de prueba en los conjuntos MNIST y Fashion-MNIST

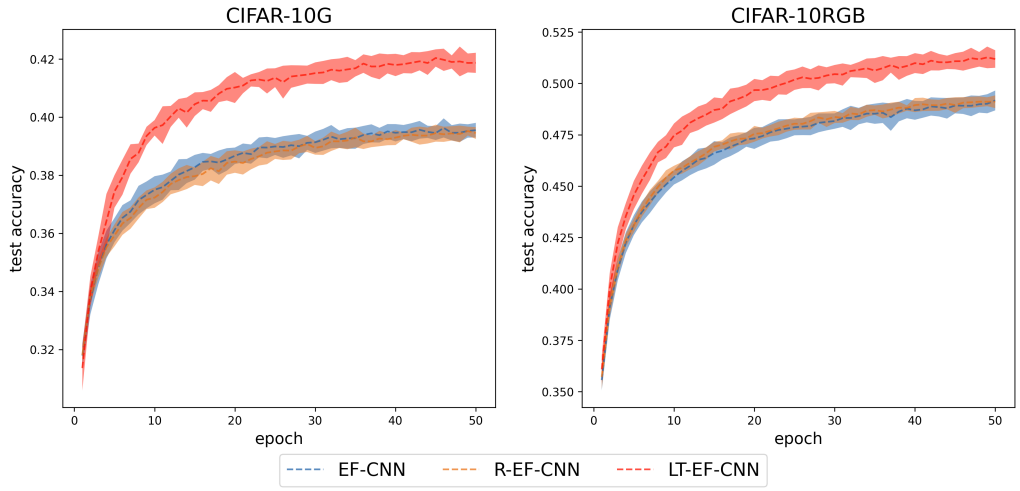
Se observa que tanto en las arquitecturas EF-CNN (Figura 5.1a) como en las SB-CNN (Figura 5.1b), la exactitud de las Redes de Fourier base y sus variantes tipo-R mantienen una exactitud similar durante todo el proceso de entrenamiento, y sus desviaciones estándar se encuentran en rangos comparables. Estos resultados nos permiten observar que, para esta base de datos, ambas arquitecturas poseen las mismas capacidades para realizar la clasificación de imágenes. Este resultado era de esperarse, dado que, como se mencionó en la Sección 4.2.1, ambas arquitecturas están trabajando con la misma cantidad de componentes frecuenciales. Por consiguiente, resulta equivalente trabajar con cualquiera de estas arquitecturas, sin embargo, las arquitecturas propuestas tipo-R ofrecen un menor costo computacional al trabajar con una representación más pequeña.

Cabe destacar que, tanto en las arquitecturas EF-CNN como en SB-CNN, las variantes tipo-LT presentan una mejora significativa en la exactitud en comparación con las arquitecturas base y tipo-R. Esto sugiere que la representación reducida construída por las capas *Linear Transform* contiene información más representativa que la representación reducida utilizada en las arquitecturas tipo-R.

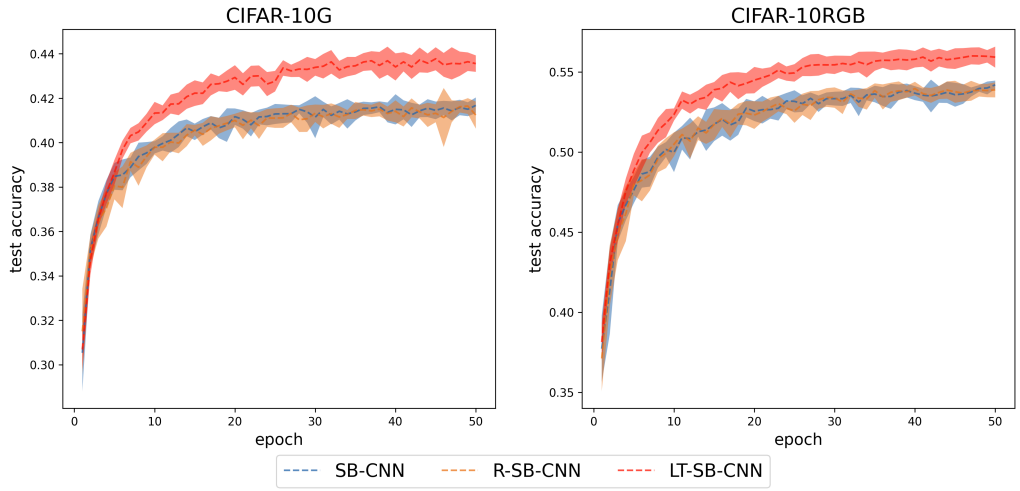
Las Figuras 5.2a y 5.2b presentan los resultados experimentales de CIFAR-10 para las arquitecturas EF-CNN y SB-CNN, respectivamente. En ambos casos se muestran los resultados obtenidos trabajando con imágenes en escala de grises y en formato RGB. Se nota que la tendencia de las arquitecturas tipo-R y tipo-LT se mantiene como se observó en los casos de MNIST y Fashion-MNIST. Sin embargo, se aprecian mejores resultados en los experimentos realizados con imágenes en RGB. Esta evidencia sugiere que se pierde información significativa al trabajar con imágenes en escala de grises en la representación frecuencial, por lo cual no se cuenta con una representación adecuada de la información para llevar a cabo la clasificación de manera óptima.

Segundo experimento

En el segundo experimento se sigue una metodología distinta. En primer lugar, se unen en un solo conjunto las particiones de entrenamiento y prueba de cada conjunto de datos, a partir del cual se generan tres nuevos subconjuntos: entrenamiento, validación y prueba, con un 70 %, 10 % y 20 % del total de datos, respectivamente. Bajo esta configuración, se realizaron 10 experimentos con las arquitecturas y condiciones experimentales descritas en la Sección 5.1.1. Cabe mencionar que no se experimentó con las arquitecturas base EF-CNN y SB-CNN, pues en el experimento anterior ya se observó su equivalencia con las arquitecturas tipo-R. En cada experimento se entreno a la red con el conjunto



(a) Arquitecturas EF-CNN, R-EF-CNN y LT-EF-CNN



(b) Arquitecturas SB-CNN, R-SB-CNN y LT-SB-CNN

Figura 5.2: Exactitud promedio en cada época en el conjunto de prueba en los conjuntos CIFAR-10G y CIFAR-10RGB

entrenamiento, posteriormente, se evaluó la exactitud de la arquitectura en el conjunto de prueba utilizando la configuración de la época que alcanzó la máxima exactitud en el conjunto de validación.

La Tabla 5.1 muestra la exactitud promedio obtenida sobre el conjunto de prueba en cada caso, con el mejor resultados en negritas. Las arquitecturas fueron evaluadas en dos configuraciones: la opción 4×4 y 8×8. Cada opción se denomina de acuerdo al tamaño de la representación que reciben las capas de neuronas completamente conectadas para realizar la clasificación (antes de la capa *Flatten*). Es importante destacar que las arquitecturas 4×4 corresponden a las mismas utilizadas en el experimento anterior, descritas en la Sección 5.1.1. Por otra parte, las arquitecturas 8×8 presentan ligeras modificaciones. En el caso de las arquitecturas tipo-R, éstas reciben una representación reducida de tamaño

8×8, mientras que las arquitecturas tipo-LT construyen una representación reducida de tamaño 8×8 con la capa *Linear Transform*. En lo que respecta a las arquitecturas CNN, se eliminó la última capa de *pooling* para lograr una representación de dicho tamaño.

Tabla 5.1: Exactitud promedio en el conjunto de prueba obtenida en cada conjunto de datos con las Redes Fourier EF-CNN y SB-CNN en sus variantes tipo-R, tipo-LT y las arquitecturas CNN, en sus versiones 4×4 y 8×8

Dataset	Arquitectura	4×4	8×8
MNIST	R-EF-CNN	$0{,}938 \pm 0{,}001$	$0{,}972 \pm 0{,}001$
	LT-EF-CNN	$0{,}965 \pm 0{,}001$	$0{,}974 \pm 0{,}001$
	R-SB-CNN	$0{,}956 \pm 0{,}001$	$0{,}978 \pm 0{,}001$
	LT-SB-CNN	$0{,}973 \pm 0{,}001$	$0{,}977 \pm 0{,}001$
	CNN	$\mathbf{0{,}988 \pm 0{,}001}$	$\mathbf{0{,}989 \pm 0{,}000}$
Fashion-MNIST	R-EF-CNN	$0{,}841 \pm 0{,}001$	$0{,}877 \pm 0{,}001$
	LT-EF-CNN	$0{,}871 \pm 0{,}002$	$0{,}881 \pm 0{,}002$
	R-SB-CNN	$0{,}861 \pm 0{,}002$	$0{,}888 \pm 0{,}001$
	LT-SB-CNN	$0{,}882 \pm 0{,}002$	$0{,}889 \pm 0{,}001$
	CNN	$\mathbf{0{,}907 \pm 0{,}002}$	$\mathbf{0{,}910 \pm 0{,}002}$
CIFAR-10G	R-EF-CNN	$0{,}386 \pm 0{,}002$	$0{,}410 \pm 0{,}005$
	LT-EF-CNN	$0{,}405 \pm 0{,}003$	$0{,}427 \pm 0{,}004$
	R-SB-CNN	$0{,}406 \pm 0{,}004$	$0{,}441 \pm 0{,}003$
	LT-SB-CNN	$0{,}424 \pm 0{,}002$	$0{,}447 \pm 0{,}003$
	CNN	$\mathbf{0{,}629 \pm 0{,}011}$	$\mathbf{0{,}629 \pm 0{,}015}$
CIFAR-10RGB	R-EF-CNN	$0{,}478 \pm 0{,}004$	$0{,}488 \pm 0{,}004$
	LT-EF-CNN	$0{,}495 \pm 0{,}004$	$0{,}519 \pm 0{,}005$
	R-SB-CNN	$0{,}530 \pm 0{,}003$	$0{,}545 \pm 0{,}003$
	LT-SB-CNN	$0{,}549 \pm 0{,}005$	$0{,}560 \pm 0{,}005$
	CNN	$\mathbf{0{,}638 \pm 0{,}013}$	$\mathbf{0{,}644 \pm 0{,}015}$

En cuanto a los resultados de las arquitecturas 4×4, se aprecia una tendencia similar a la observada en el experimento anterior, donde las arquitecturas tipo-LT presentan una mejora significativa en comparación con las arquitecturas tipo-R. Además, al comparar las arquitecturas en la representación frecuencial, se observa una clara superioridad de las arquitecturas SB-CNN sobre las arquitecturas EF-CNN en todos los conjuntos de datos. Esta observación sugiere que los *kernels* aleatorios propuestos por Han et al. [11] no son capaces de identificar patrones tan complejos como los *kernels* usados en las arquitecturas SB-CNN, los cuales tienen más flexibilidad para aprender, al estar compuestos enteramente por parámetros

entrenables.

Al comparar con el enfoque convencional, se observa que las arquitecturas tipo-LT presentan una exactitud similar a CNN, excepto en el caso de CIFAR-10, lo cual indica que se pierde información importante de imágenes naturales al trabajar en la representación frecuencial. La diferencia en exactitud es más significativa en los experimentos con imágenes en escala de grises. Sin embargo, en el caso RGB, la diferencia de exactitud en comparación con CNN disminuye, lo cual refuerza la idea de que cada canal de la imagen en color contiene información importante de la imagen en la representación frecuencial. Por el contrario, esta diferencia de exactitud entre la clasificación con imágenes en escala de grises y en formato RGB no se observa en los experimentos con CNN.

En lo que respecta a los resultados experimentales de las arquitecturas 8×8, se observa una clara diferencia en la exactitud de las arquitecturas tipo-R 8×8 en comparación con sus equivalentes 4×4. Esto se atribuye a dos factores, en primer lugar, se espera que la exactitud de las arquitecturas tipo-R aumente a medida que se trabaja con representaciones reducidas más grandes, ya que se considera más información de la representación frecuencial. Por otro lado, cuanto mayor sea el tamaño de la representación reducida, mayor será el tamaño de los *kernels* en la representación frecuencial, lo que permite una mayor variedad de configuraciones para identificar patrones complejos en las imágenes. También es importante destacar que las arquitecturas tipo-LT 8×8 no presentan una ventaja considerable con sus equivalentes tipo-R, como en el caso de las arquitecturas 4×4. Esto se debe a la cantidad de información considerada en cada caso. Por ejemplo, en el caso de MNIST y Fashion MNIST, con una representación de tamaño 28×28, una representación reducida de tamaño 4×4 solo constituye el $2\,\%$ de la información total, mientras que una representación reducida de tamaño 8×8 representa el $8\,\%$. Por esta razón la capa *Linear Transform* es de gran ayuda en el primer caso, donde se trabaja con una porción muy pequeña de toda la información disponible, mientras que en el segundo caso, el aporte de la capa no es tan significativo, ya que la representación reducida contiene suficiente información. Aunque es evidente la superioridad de las arquitecturas 8×8 en comparación con las arquitecturas 4×4, también se observa que las arquitecturas tipo-LT 4×4 consiguen resultados similares a las arquitecturas tipo-R 8×8, lo que indica que la capa *Linear Transform* es capaz de construir una representación de la información tan relevante como en el caso de representaciones reducidas más grandes. Por lo tanto, trabajar con una arquitectura tipo-LT 4×4 puede ser más conveniente que con una arquitectura tipo-R 8×8, ya que la segunda opción es mucho más costosa computacionalmente, y solo proporciona una ligera mejora en la exactitud.

5.1.3. Evaluación de número de parámetros y operaciones de punto flotante

En esta sección se compara el costo computacional de las arquitecturas descritas en la Sección 5.1.1. Para esto, se realiza una comparación del número de parámetros de cada arquitectura y el número de operaciones de punto flotante (FLOP) realizadas en la inferencia de una única instancia. Para CNN y las Redes de Fourier base, se considera una representación de entrada de dimensión 28×28, correspondiente al tamaño de los conjuntos de datos MNIST y Fashion-MNIST. Para las arquitecturas propuestas, se analiza el caso de las arquitecturas tipo-R, considerando una representación reducida de tamaño 4×4. En esta sección el análisis se centra sobre las arquitecturas tipo-R, ya que las tipo-LT presentan resultados similares, pues comparten la misma estructura, con la única diferencia de que contienen además la capa *Linear Transform*, la cual involucra un incremento de $3,136$ parámetros y $12,544$ operaciones de punto flotante, las cuales no son significativas para el análisis general. Adicionalmente, se incluye un análisis del costo computacional de las arquitecturas 8×8 utilizadas en los experimentos de la Sección 5.1.2.

Arquitecturas base

Las Tablas 5.2 y 5.3 reportan el recuento de parámetros y operaciones de punto flotante de cada arquitectura, respectivamente, con el mejor resultados en negritas. Asimismo, las Tablas 5.4 y 5.5 reportan el factor de reducción logrado por las arquitecturas tipo-R en comparación con CNN y las Redes de Fourier base. Este factor se calcula como el cociente entre la medida de una arquitectura de referencia y la medida de la arquitectura tipo-R. De esta manera, los factores de reducción con valores superiores a 1 reflejan una reducción, mientras que los menores a 1 indican un aumento. En cada tabla, cada fila compara los resultados de cada capa involucrada en el análisis, y la última fila recaba el total de los resultados. Note que solo se comparan las capas previas a la capa *Flatten*, ya que posteriormente todas las arquitecturas presentan la misma configuración, y por lo tanto el mismo número de parámetros y operaciones de punto flotante.

En lo que respecta al número de parámetros, se observa una reducción significativa al comparar las arquitecturas tipo-R con las Redes de Fourier base, siendo esta reducción es más notable en las arquitecturas EF-CNN. Cabe notar que esta reducción no es tan evidente en el caso de las arquitecturas SB-CNN, debido a la implementación de la estrategia descrita por Ayat et al. [1] (ver Sección 3.5) para ahorrar parámetros al realizar la convolución. Aún así, se aprecia una disminución en el número de parámetros, especialmente en las primeras capas convolucionales.

Tabla 5.2: Número de parámetros por capa utilizando un conjunto de datos con un tamaño de entrada de 28×28 para CNN, EF-CNN y SB-CNN, y una representación reducida de tamaño 4×4 para R-EF-CNN y R-SB-CNN

Capa	CNN	EF-CNN	SB-CNN	R-EF-CNN	R-SB-CNN
Convolución 1	**156**	9,420	2,364	204	204
Convolución 2	**2,416**	37,824	9,440	3,264	3,104
Convolución 3	**48,120**	192,000	61,680	65,280	61,680
Total	**50,692**	239,244	73,484	68,748	64,988

Tabla 5.3: Número de operaciones de punto flotante en inferencia por capa, utilizando un conjunto de datos con un tamaño de entrada de 28×28 para CNN, EF-CNN y SB-CNN, y una representación reducida de tamaño 4×4 para R-EF-CNN y R-SB-CNN

Capa	CNN	EF-CNN	SB-CNN	R-EF-CNN	R-SB-CNN
FFT	-	15,076	15,076	15,076	15,076
Convolución 1	239,904	37,632	9,408	**768**	**768**
Pooling 1	4,704	0	0	-	-
Convolución 2	943,936	181,888	37,632	14,848	**12,288**
Pooling 2	3,136	0	0	-	-
Convolución 3	4,709,880	929,040	245,760	303,360	**245,760**
Pooling 3	7,680	0	0	-	-
Transformación	-	5,760	41,882	5,760	41,882
Total	5,909,240	1,169,396	349,758	339,812	**315,774**

Los resultados de la Tabla 5.4 muestran un alto factor de reducción entre las arquitecturas tipo-R y sus equivalente Redes de Fourier base, lo que conlleva a un gran ahorro en parámetros, principalmente en las primeras capas convolucionales, donde la representación es más grande y, por ende, requiere un mayor número de parámetros. Al comparar con CNN, se observa que las arquitecturas tipo-R consiguen un número de parámetros similar. Sin embargo, aún existe margen de mejora, como lo indican los resultados reportados en la Tabla 5.4, los cuales muestran que el número de parámetros de CNN representan el 73,7 % y 78,0 % de los parámetros de las arquitecturas R-EF-CNN y R-SB-CNN, respectivamente.

Note que al comparar las arquitecturas tipo-R con sus arquitecturas base, el factor de reducción decrece a medida que la arquitectura se hace más profunda, por el contrario, el factor de reducción se mantiene fijo al comparar con CNN. Esto se debe a que, tanto en las arquitecturas tipo-R como en CNN, el número de

Tabla 5.4: Reducción en parámetros de las arquitecturas tipo-R utilizando una representación reducida de tamaño 4×4, en comparación con CNN y Redes de Fourier base, con un tamaño de entrada de 28×28

Capa	R-EF-CNN			R-SB-CNN		
	CNN	EF-CNN	SB-CNN	CNN	EF-CNN	SB-CNN
Convolución 1	×0,765	×46,176	×11,588	×0,765	×46,176	×11,588
Convolución 2	×0,740	×11,588	×2,892	×0,778	×12,186	×3,041
Convolución 3	×0,737	×2,941	×0,945	×0,780	×3,113	×1,000
Total	×0,737	×3,480	×1,069	×0,780	×3,681	×1,131

Tabla 5.5: Reducción en operaciones de punto flotante de las arquitecturas tipo-R utilizando una representación reducida de tamaño 4×4, en comparación con CNN y Redes de Fourier base, con un tamaño de entrada de 28×28

Capa	R-EF-CNN			R-SB-CNN		
	CNN	EF-CNN	SB-CNN	CNN	EF-CNN	SB-CNN
Convolución 1	×312,375	×49,000	×12,250	×312,375	×49,000	×12,250
Convolución 2	×63,573	×12,250	×2,534	×76,818	×14,802	×3,062
Convolución 3	×15,526	×3,062	×0,810	×19,165	×3,780	×1,000
Total	×17,390	×3,441	×1,029	×18,714	×3,703	×1,108

parámetros por *kernel* se mantiene constante en todas las capas, mientras que en las Redes de Fourier base, el número de parámetros depende del tamaño de la representación frecuencial de entrada a cada capa.

En cuanto al número de operaciones de punto flotante en inferencia, los resultados presentados en la Tabla 5.3 revelan una reducción significativa al comparar las arquitecturas tipo-R con sus arquitecturas base equivalente y CNN, siendo esta diferencia más notable en el último caso. La Tabla 5.5 muestra que las arquitecturas tipo-R logran reducir el total del número de operaciones en comparación con sus arquitecturas base EF-CNN y SB-CNN, en un factor de ×4,480 y ×1,131, respectivamente. En el último caso, aunque el factor de reducción total no es tan significativo debido a la técnica de optimización implementada, se observa una reducción considerable en las primeras capas convolucionales, donde falla la estrategia propuesta por Ayat et al. [1]. Asimismo, se observa un alto factor de reducción en las capas convolucionales al comprar las arquitecturas tipo-R con CNN, con resultados superiores a ×300 en la primera capa convolucional tanto en R-EF-CNN como en R-SB-CNN, y logrando un factor de reducción total de ×17,390 y ×18,714, respectivamente.

Note que en este caso, el factor de reducción entre las arquitecturas tipo-R en comparación con las Redes de Fourier base y CNN decrece conforme se hace más profunda la arquitectura. Esto se debe a que las arquitecturas tipo-R presentan la cualidad de que el número de operaciones por cada canal de cada filtro se mantiene constante. Por lo cual, estas arquitecturas permiten manejar un gran número de filtros, incluso en las primeras capas convolucionales, donde una CNN convencional tendría dificultades para incorporar una cantidad elevada de filtros.

Considerando los resultados presentados, las arquitecturas propuestas tipo-R se colocan como una alternativa más viable que sus arquitecturas base EF-CNN y SB-CNN, ya que presentan un menor costo computacional tanto en el número de parámetros como en las operaciones de punto flotante en inferencia, manteniendo al mismo tiempo el mismo nivel de exactitud, como se demostró en la Sección 5.1.2. Además, al considerar una capa *Linear Transform* para construir una arquitecturas tipo-LT, se consigue un aumento en la exactitud, sin incurrir en un alto costo computacional adicional. Considerando una capa *Linear Transform* de salida 4×4, una arquitectura LT-EF-CNN tiene un total de $71,884$ parámetros y $352,356$ operaciones, mientras que la LT-SB-CNN cuenta con $68,124$ parámetros y $328,318$ operaciones. En consecuencia, el coste computacional sigue siendo inferior al de las arquitecturas base y las CNN convencionales, con la ventaja de mejorar la exactitud en comparación con las arquitecturas tipo-R.

Arquitecturas 8×8

Considerando una representación reducida de tamaño 4×4, las arquitecturas propuestas ofrecen una opción viable en términos de costo computacional frente a las arquitecturas CNN. Sin embargo, al trabajar con representaciones reducidas más grandes, como las arquitecturas tipo-R 8×8 usadas en los experimentos, se observa un gran incremento en el número de parámetros y operaciones de punto flotante, como se reporta en las Tablas 5.6 y 5.7.

Tabla 5.6: Número de parámetros por capa utilizando un conjunto de datos con un tamaño de entrada de 28×28 para CNN, y una representación reducida de tamaño 8×8 para R-EF-CNN y R-SB-CNN

Capa	CNN	R-EF-CNN	R-SB-CNN
Convolución 1	**156**	816	816
Convolución 2	**2,416**	13,056	12,416
Convolución 3	**48,120**	261,120	246,720
Total	**50,692**	274,992	259,952

Tabla 5.7: Número de operaciones de punto flotante en inferencia por capa utilizando un conjunto de datos con un tamaño de entrada de 28×28 para CNN, y una representación reducida de tamaño 8×8 para R-EF-CNN y R-SB-CNN

Capa	CNN	R-EF-CNN	R-SB-CNN
FFT	-	15,076	15,076
Convolución 1	239,904	**3,072**	**3,072**
Pooling 1	4,704	-	-
Convolución 2	943,936	59,392	**49,152**
Pooling 2	3,136	-	-
Convolución 3	4,709,880	1,213,440	**983,040**
Transformación	-	23,040	198,249
Total	5,901,560	1,314,020	**1,248,589**

Es importante notar que tanto la arquitectura R-EF-CNN como R-SB-CNN muestran un incremento de factor $\times 4$ en el número de parámetros y operaciones de punto flotante en las capas convolucionales en comparación con las arquitecturas 4×4.

Incluso bajo estas condiciones, las arquitecturas tipo-R 8×8 realizan menos operaciones de punto flotante en inferencia que las arquitecturas CNN. Sin embargo, el número de parámetros no es comparable con CNN. Este problema se vuelve más evidente al trabajar con imágenes de mayor tamaño, donde se requiere una representación reducida más grande, lo que podría limitar su implementación en dispositivos con recursos de memoria limitados debido al aumento en la cantidad de parámetros por *kernel* en estos casos.

5.2. Enfoque para imágenes grandes

En esta sección se evalúa el desempeño de las arquitecturas propuestas BW-CNN al trabajar con conjuntos de imágenes grandes. Se examina, tanto la exactitud de las arquitecturas como su costo computacional, comparadas con arquitecturas CNN y con arquitecturas tipo-R.

5.2.1. Condiciones experimentales

Para evaluar a las arquitecturas BW-CNN, se eligieron los conjuntos de datos: *Colorectal-Cancer Histology* (comúnmente denominado HMNIST en la literatura) [36] y *Human Against Machine with 10000 training images* (HAM10000) [37]. Ambos conjuntos de datos están formados por imágenes médicas grandes. HMNIST

consiste en patrones histológicos de posibles pacientes con cáncer colorectal, con 5,000 imágenes en formato RGB de resolución 150×150, distribuidas en 8 posibles clases de patologías, incluyendo el cáncer. Al igual que los conjuntos de datos usados en la sección anterior, HMNIST presenta clases perfectamente balanceadas, pero sus patrones son más complejos de identificar, lo que lo hace ideal para evaluar las arquitecturas propuestas. Por otro lado, el conjunto de datos HAM10000 contiene imágenes de 7 posibles lesiones en la piel, con un total de 10,015 imágenes en formato RGB de tamaño 600×450. A diferencia de HMNIST, HAM10000 presenta el problema de clases desbalanceadas, como se aprecia en la Tabla 5.8, que muestra el total de instancias por clase.

Tabla 5.8: Número de instancias por clase en el conjunto de datos HAM10000

Clase	Instancias
akiec	327
bcc	514
bkl	1,099
df	115
mel	1,113
nv	6,705
vasc	142

Para trabajar con estas imágenes, ambos conjuntos se redimensionaron a una resolución de 128×128. Los experimentos se realizaron tanto en escala de grises como en formato RGB, como se describió en la Sección 5.1.1. Como arquitecturas base para la evaluación, se consideraron una CNN convencional y una arquitectura R-SB-CNN, dado que, como se demostró en la sección anterior, las arquitecturas SB-CNN demuestran resultados superiores a las arquitecturas EF-CNN. En cuanto a las arquitecturas propuestas BW-CNN, estas incluyen la estrategia de filtros Butterworth (ver Sección 4.3.1) para generar *kernels* de manera dinámica al realizar la convolución, además de emplear la capa de *pooling* propuesta, *Spectral Average Pooling* (ver Sección 4.3.2). Siguiendo el enfoque de las arquitecturas propuestas por Ayat et al. [1], se optó por la transformada de Fourier inversa como transformación de coordenadas para la representación compleja. A continuación, se proporciona una descripción detallada de cada una de estas arquitecturas:

- **CNN:** esta arquitectura recibe como entrada a las imágenes en su representación espacial. Contiene 3 capas convolucionales de 8, 32 y 64 filtros de tamaño 5×5 respectivamente, evaluadas por la función de

activación ReLU y una capa *Max Pooling*. Después de la tercera capa de *Max Pooling*, se incluye una capa *Flatten* que conecta con una capa de *Batch Normalization*. Luego, se incluyen dos capas de neuronas completamente conectadas: la primera de 128 neuronas con función de activación ReLU, y la última con función de activación softmax. Dependiendo del conjunto de datos, la última capa tendrá 8 neuronas para HMNIST y 7 neuronas para HAM10000.

- **R-SB-CNN:** en este caso la arquitectura recibe una representación reducida de tamaño 16×16, construida por las componentes frecuenciales centrales de la transformada de Fourier centrada (ver Sección 2.2.3). Se mantiene la misma disposición de capas convolucionales que en la arquitectura CNN, realizando la operación de convolución como se describió en la Sección 4.1 y con todas las componentes de los *kernels* como parámetros entrenables. La función de activación empleada es CReLU (consultar Ecuación 3.3), y se omite el uso de capas de *pooling*. Después de la tercera capa convolucional, la arquitectura calcula la transformada rápida de Fourier inversa de la representación compleja. El resto de la configuración es idéntica a la de la arquitectura CNN.

- **BW-CNN:** a diferencia de R-SB-CNN, la arquitectura BW-CNN recibe como entrada toda la representación frecuencial centrada. En lo que respecta a las capas convolucionales, se mantiene la misma configuración que CNN y R-SB-CNN. La operación de convolución se realiza como se describe en la Sección 4.1, utilizando la técnica de filtros Butterworth. La configuración de parámetros usados para implementar esta técnica es la siguiente (ver Sección 4.3.1): $d = 2$, $g_0 = 0{,}45$, $\mathrm{dim} = 1{,}0$ y $R = 0{,}25$. Cada capa convolucional se evalúa con la función de activación CReLU, seguida de la capa *Spectral Average Pooling*. Similar a R-SB-CNN, después de la tercera capa de *pooling*, se calcula transformada de Fourier inversa. El resto de la configuración se mantiene como en las arquitecturas CNN y R-SB-CNN.

Tanto la arquitectura CNN como BW-CNN presentan una representación de tamaño 16×16 después de la tercera capa de *pooling*, por lo cual la representación reducida de la arquitectura R-SB-CNN posee este mismo tamaño. De esta manera, todas las arquitecturas cuentan con la misma cantidad de neuronas e información para llevar a cabo la clasificación de las imágenes, lo que garantiza una comparación en igualdad de condiciones. Todos los experimentos de esta sección se realizaron entrenando durante 50 épocas, con un optimizador SGD con coeficiente de aprendizaje 1×10^{-5}, *Momentum* 0,9 y tamaño de *batch* 32.

5.2.2. Evaluación de exactitud

Se llevaron a cabo dos experimentos para evaluar la exactitud de las arquitecturas propuestas BW-CNN. Ambos experimentos siguieron una configuración similar a la descrita en la Sección 5.1.2, dividiendo el conjunto en 3 subconjuntos: entrenamiento, prueba y validación, cada uno con 70 %, 10 % y 20 % del total de los datos, respectivamente. El entrenamiento se realiza sobre el conjunto de entrenamiento, mientras que se monitorea el desempeño en cada época sobre el conjunto de validación. Finalmente, se evaluó el rendimiento de cada arquitectura en el conjunto de prueba utilizando la configuración de la época que proporcionó la exactitud máxima en el conjunto validación.

El primer experimento se realizó con el conjunto de datos HMNIST, el propósito de este experimento es evaluar la exactitud obtenida por BW-CNN bajo condiciones ideales. Por otro lado, el segundo experimento se realizó con el conjunto de datos HAM10000, en este caso se busca evaluar a la arquitectura BW-CNN al enfrentarse al problema de clases desbalanceadas. También se evalúa a la arquitectura construyendo un subconjunto de clases balanceadas, seleccionando 115 instancias aleatorias por clase. Dado que, como se observa en la Tabla 5.8, la clase con menos datos (df) contiene 115 instancias.

Experimentos con HMNIST

La Figura 5.3 y la Tabla 5.9 presentan los resultados experimentales obtenidos en el conjunto de datos HMNIST, tanto en escala de grises (denotado como HMNIST-G) como en formato RGB (denotado como HMNIST-RGB). La Figura 5.3 muestra el comportamiento de cada arquitectura durante el entrenamiento. En cada caso se muestra la exactitud promedio obtenida sobre el conjunto validación en cada época, con la desviación estándar representada por la región sombreada. Por otro lado, la Tabla 5.9 muestra la exactitud obtenida al evaluar al conjunto de prueba.

En lo que respecta a los resultados con HMNIST-G, la Figura 5.3 muestra claramente que la arquitectura BW-CNN tiende a exhibir una exactitud superior al resto de arquitecturas durante todo el entrenamiento, seguida por SB-CNN y, finalmente CNN. Esta tendencia se confirma al evaluar a cada arquitectura en el conjunto de prueba (Tabla 5.9), donde se observa una clara ventaja de las arquitecturas BW-CNN sobre las demás. Sin embargo, en este caso CNN logra resultados superiores a R-SB-CNN.

Por otro lado, en el entrenamiento sobre imágenes en formato RGB (HMNIST-RGB), la Figura 5.3 indica que la arquitecturas R-SB-CNN mantienen una exactitud superior a al resto durante todo el entrenamiento, mientras que

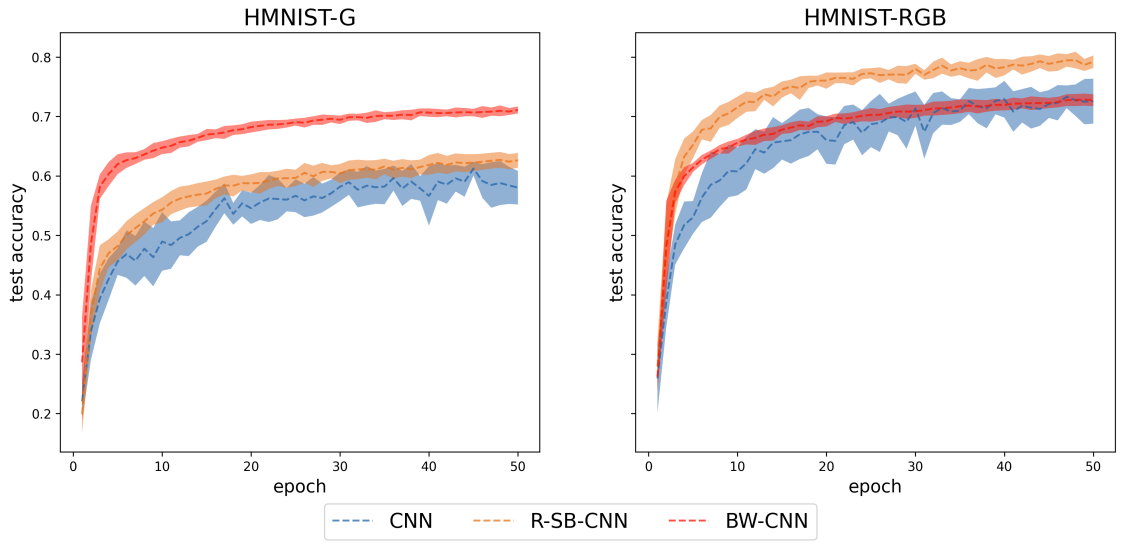


Figura 5.3: Exactitud promedio al evaluar el conjunto validación en cada época con HMNIST en escala de grises y en formato RGB

Tabla 5.9: Exactitud promedio obtenida en el conjunto de prueba con HMNIST en escala de grises y en formato RGB

Dataset	Arquitectura	exactitud
HMNIST-G	CNN	$0{,}640 \pm 0{,}012$
	R-SB-CNN	$0{,}624 \pm 0{,}014$
	BW-CNN	$\mathbf{0{,}706 \pm 0{,}010}$
HMNIST-RGB	CNN	$0{,}720 \pm 0{,}023$
	R-SB-CNN	$\mathbf{0{,}791 \pm 0{,}010}$
	BW-CNN	$0{,}739 \pm 0{,}011$

BW-CNN y CNN muestran una exactitud similar. La Tabla 5.9 confirma esta tendencia, ya que la arquitecturas R-SB-CNN presenta claramente una exactitud superior al resto, seguidas de las arquitecturas BW-CNN y por último CNN.

En general, los resultados presentados en la Tabla 5.9 muestran que existe una diferencia considerable en la exactitud de cada arquitectura al trabajar con imágenes en escala de grises y en formato RGB, siendo más precisa esta última opción.

Cabe notar que en cada caso, las arquitecturas que consiguen la mejor exactitud son aquellas que trabajan en la representación frecuencial, lo que sugiere que este enfoque es efectivo para manejar conjuntos de datos complejos como las imágenes médicas. En general se observa que la arquitecturas BW-CNN presenta mejor exactitud que CNN. Sin embargo, la arquitectura R-SB-CNN también presenta buenos resultados, principalmente al trabajar con imágenes

en formato RGB, donde R-SB-CNN consigue mejores resultados que BW-CNN, probablemente debido al tamaño de los *kernels*, los cuales, aunque son capaces de identificar patrones complejos para mejorar la exactitud, también involucran un número elevado de parámetros, dado que su tamaño coincide con el de la representación reducida, de dimensión 16×16.

Experimentos con HAM10000

La Figura 5.4 y la Tabla 5.10 muestran los resultados experiméntales obtenidos al trabajar con el conjunto de datos HAM10000 en escala de grises (denotado como HAM10000-G) y en formato RGB (denotado como HAM10000-RGB). Los resultados presentados en la Tabla 5.10 incluyen tanto la métrica de exactitud, como las métricas precisión, recuerdo y F1-score, en estas últimas se considera el promedio de la evaluación individual de cada métricas sobre cada clase, en la literatura este enfoque se denomina "macro". En la Figura 5.4 se observa que, en el caso de imágenes en escala de grises, ambas arquitecturas en la representación frecuencial R-SB-CNN y BW-CNN presentan un desempeño similar durante todo el entrenamiento, mientras que CNN presenta una exactitud muy variable. Esto se atribuye principalmente al hecho de entrenar con imágenes en escala de grises, ya que se cuenta con menos información disponible para realizar una clasificación adecuada. La Tabla 5.4 muestra que, al evaluar a las arquitecturas sobre el conjunto de prueba, todas presentan una exactitud similar, con una ligera ventaja de R-SB-CNN.

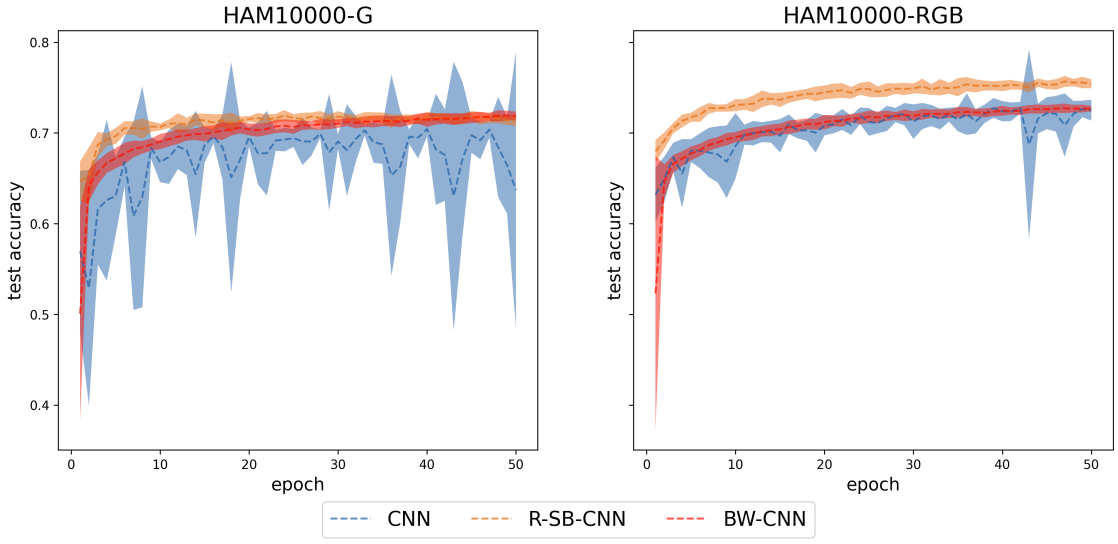


Figura 5.4: Exactitud promedio al evaluar el conjunto validación en cada época con HAM10000 en escala de grises y en formato RGB

De manera similar se observa que, al trabajar con el conjunto de datos en formato RGB (HAM10000-RGB), las arquitecturas BW-CNN y CNN presentan un

Tabla 5.10: Precisión, recuerdo, F1-score y exactitud promedio obtenida en el conjunto prueba con HAM10000 en escala de grises y en formato RGB

Dataset	Arquitectura	precisión	recuerdo	F1-score	exactitud
HAM10000-G	CNN	**0,342 ± 0,061**	0,245 ± 0,018	0,259 ± 0,017	0,695 ± 0,007
	R-SB-CNN	0,326 ± 0,009	**0,255 ± 0,010**	**0,273 ± 0,009**	**0,697 ± 0,004**
	BW-CNN	0,322 ± 0,012	0,249 ± 0,006	0,268 ± 0,007	0,693 ± 0,004
HAM10000-RGB	CNN	0,425 ± 0,055	0,329 ± 0,020	0,348 ± 0,023	0,715 ± 0,008
	R-SB-CNN	**0,449 ± 0,041**	**0,353 ± 0,020**	**0,379 ± 0,022**	**0,726 ± 0,004**
	BW-CNN	0,336 ± 0,010	0,266 ± 0,007	0,287 ± 0,008	0,702 ± 0,003

desempeño similar durante todo el entrenamiento, mientras que la arquitectura R-SB-CNN presenta una ligera ventaja. Además, los resultados presentados en la Tabla 5.10 revelan que, al evaluar cada arquitectura sobre el conjunto prueba, no hay una gran diferencia en la exactitud entre todas las arquitecturas.

En términos generales, los resultados son similares para todas las arquitecturas, tanto en escala de grises como en formato RGB. Este hecho se atribuye al problema de clases desbalanceadas, ya que, como revela la Tabla 5.8, la clase nv posee el 66,94 % del total de instancias en el conjunto de datos. La Tabla 5.10 reporta un bajo desempeño en las métricas precisión y recuerdo, lo que indica una alta tasa de falsos positivos y falsos negativos. En consecuencia, el F1-score obtenido es bajo. Estos resultados sugieren que las predicciones realizadas por las arquitecturas son poco confiables.

Por otro lado, la Tabla 5.11 muestra la exactitud promedio obtenida sobre el conjunto prueba al trabajar con el conjunto de datos HAM10000 balanceado, tanto en escala de grises (denotado como HAM10000*-G) como en RGB (denotado como HAM10000*-RGB). En contraste con los resultados presentados usando todo el conjunto de datos, se observa que en este caso si existe una notable ventaja al trabajar con imágenes en formato RGB. No obstante, se observa que todas las arquitecturas presentan un decremento considerable en la exactitud en comparación con los resultados obtenidos al trabajar con todos los datos. Este problema se atribuye al reducido número de instancias consideradas.

En los trabajos reportados en la literatura, este conjunto de datos ha sido ampliamente examinado, consiguiendo resultados con exactitud superior a 80 %, como es el caso del trabajo de Ali et al. [38], donde se alcanza una exactitud del 87,91 %. El enfoque seguido por los autores consiste en una técnica de aumento de datos para abordar el inconveniente de clases desbalanceadas, además de utilizar arquitecturas CNN profundas. En particular, su mejor resultado es obtenido con una arquitectura EfficientNet B4 [39], que consta de 19 millones de parámetros, en comparación con 2,222,152, 3,355,992 y 2,173,160 parámetros

Tabla 5.11: Exactitud promedio obtenida en el conjunto prueba con el conjunto de datos HAM10000 balanceado en escala de grises y en formato RGB

Dataset	Arquitectura	exactitud
HAM10000*-G	CNN	$0{,}252 \pm 0{,}034$
	R-SB-CNN	$0{,}293 \pm 0{,}018$
	BW-CNN	$\mathbf{0{,}294 \pm 0{,}025}$
HAM10000*-RGB	CNN	$0{,}363 \pm 0{,}069$
	R-SB-CNN	$\mathbf{0{,}454 \pm 0{,}040}$
	BW-CNN	$0{,}325 \pm 0{,}028$

de las arquitecturas CNN, R-SB-CNN y BW-CNN, respectivamente. Diferentes trabajos han obtenido resultados similares, como se observa en [40, 41, 42, 43]. En general, estos estudios emplean diversas técnicas como el aumento de datos o el aprendizaje por transferencia para hacer frente al problema de clases desbalanceadas. Desafortunadamente estas técnicas aún no se han explorado en el contexto de Redes de Fourier. Otra peculiaridad que comparten estos trabajos es el uso de arquitecturas de redes neuronales profundas. Este tipo de redes son fáciles de implementar con arquitecturas tipo CNN, ya que requieren un bajo número de parámetros por filtro. En el dominio frecuencial, estas arquitecturas pueden replicarse tanto con las BW-CNN como con las de tipo-R, aunque estas últimas presentan desafíos de memoria que dificultan su aplicación, como se analiza en la siguiente sección.

5.2.3. Número de parámetros y operaciones de punto flotante

En esta sección se presenta una comparación del costo computacional de las arquitecturas BW-CNN, CNN y R-SB-CNN, considerando la configuración descrita en la Sección 5.2.1. Para BW-CNN y CNN, se considera una representación de entrada 128×128, compleja en el caso de BW-CNN y espacial para CNN, mientras que para R-SB-CNN se considera una representación compleja de dimensión 16×16. Note que estas dimensiones coinciden con las utilizadas en los experimentos con HMNIST y HAM10000.

Las Tablas 5.12 y 5.14 muestran el número de operaciones de punto flotante realizadas en inferencia por una instancia y el número de parámetros de cada arquitectura, respectivamente, resaltando los mejores resultados en negritas. Note que solo se incluyen las capas previas a la capa *Flatten*, ya que posteriormente todas las arquitecturas presentan la misma configuración. Por otro lado, las Tablas 5.13 y 5.15 presentan el factor de reducción en operaciones y parámetros,

Tabla 5.12: Número de operaciones de punto flotante en inferencia por capa, utilizando un conjunto de datos con un tamaño de entrada de 128×128 para CNN, y BW-CNN, y una representación reducida de tamaño 16×16 para R-SB-CNN

Capa	CNN	R-SB-CNN	BW-CNN
FFT	**0**	458 752	458 752
Convolución 1	19 791 872	**49 152**	5 537 792
Pooling 1	131 072	**0**	262 144
Convolución 2	52 559 872	**524 288**	14 688 256
Pooling 2	131 072	**0**	262 144
Convolución 3	104 923 136	**4 194 304**	29 362 176
Pooling 3	65 536	**0**	131 072
Transfromación	**0**	458 752	458 752
Total	177 602 560	**5 685 248**	51 161 088

Tabla 5.13: Reducción en operaciones de punto flotante de las arquitecturas BW-CNN en comparación con CNN y R-SB-CNN

Capa	CNN	R-SB-CNN
Convolución 1	$\times$3,574	$\times$0,009
Pooling 1	$\times$0,500	-
Convolución 2	$\times$3,578	$\times$0,036
Pooling 2	$\times$0,500	-
Convolución 3	$\times$3,573	$\times$0,143
Pooling 3	$\times$0,500	-
Total	$\times$3,471	$\times$0,111

respectivamente. Este factor se calcula como la medida de una arquitectura base (CNN o R-SB-CNN) sobre la medida de la arquitectura BW-CNN.

Operaciones en inferencia

Los resultados de las Tablas 5.12 y 5.13 revelan que, al comparar las operaciones efectuadas por las capas de *pooling* en las arquitecturas CNN y BW-CNN, las capas *Max Pooling* realizan la mitad de operaciones que la capa propuesta *Spectral Average Pooling*. Esto era de esperarse, dado que ambas capas realizan una transformación similar por ventanas, pero *Spectral Average Pooling* realiza el doble de transformaciones al trabajar por separado con la parte real e imaginaria de la representación frecuencial (ver Sección 4.3.2). Sin embargo, a pesar de esta disparidad, las arquitecturas BW-CNN en general realizan menos

operaciones que CNN, con un factor de reducción total de $\times$3,471.

Por otro lado, al comparar el número de operaciones de punto flotante realizadas en capas convolucionales, se observa que la arquitectura BW-CNN realiza menos operaciones que CNN, consiguiendo un factor de reducción de alrededor de $\times$3,5 en todas las capas convolucionales. Sin embargo, la arquitectura R-SB-CNN realiza considerablemente menos operaciones que la arquitectura BW-CNN, principalmente en la primera capa convolucional, donde la arquitectura R-SB-CNN realiza solo el 0,9 % del total de operaciones realizadas por BW-CNN. Esta marcada diferencia se atribuye a las características de la arquitectura R-SB-CNN, ya que todas sus capas convolucionales operan sobre una representación de dimensión 16×16, en contraste con las arquitecturas CNN y BW-CNN, cuyas capas convolucionales reciben una representación de tamaño 128×128, 64×64 y 32×32.

Cabe mencionar que el mecanismo de filtros Butterworth implementado en la convolución de las arquitecturas BW-CNN requiere la evaluación de la función generadora $G(m, n)$ (ver Ecuación 4.6) una vez por cada *batch* a evaluar, independientemente de su tamaño. Por consiguiente, el costo computacional asociado a esta evaluación es significativo cuando se evalúan pocas instancias, como sucede en los resultados presentados en las Tablas 5.12 y 5.13, los cuales consideran el peor caso, evaluar una única instancia. La Figura 5.5 muestra el número de operaciones de punto flotante en escala logarítmica realizadas en las 3 capas convolucionales, considerando diferentes tamaños de *batch*. A medida que se consideran más instancias en el *batch*, mayor es la diferencia entre el número de operaciones realizadas por la arquitectura CNN y las arquitecturas en la representación frecuencial (BW-CNN y R-SB-CNN). Además, note que a medida que la arquitectura es más profunda, menor es la diferencia de operaciones entre BW-CNN y R-SB-CNN. Esto se atribuye a la diferencia de tamaño de las representaciones con las que operan las capas convolucionales. Por lo cual, la segunda y tercera capa convolucional realizan un número de operaciones en un orden similar.

En general, se observa que R-SB-CNN es la arquitectura que requiere el menor número de operaciones de punto flotante. Sin embargo, a medida que se incrementa el número de instancias en el *batch*, tanto BW-CNN como R-SB-CNN mantienen un número de operaciones significativamente menor al de CNN. Por lo tanto, ambas arquitecturas en la representación frecuencial presentan una mejora sustancial en comparación con el enfoque convencional de CNN.

Número de parámetros

Sin embargo, como se observa en la Tablas 5.14 las arquitecturas R-SB-CNN

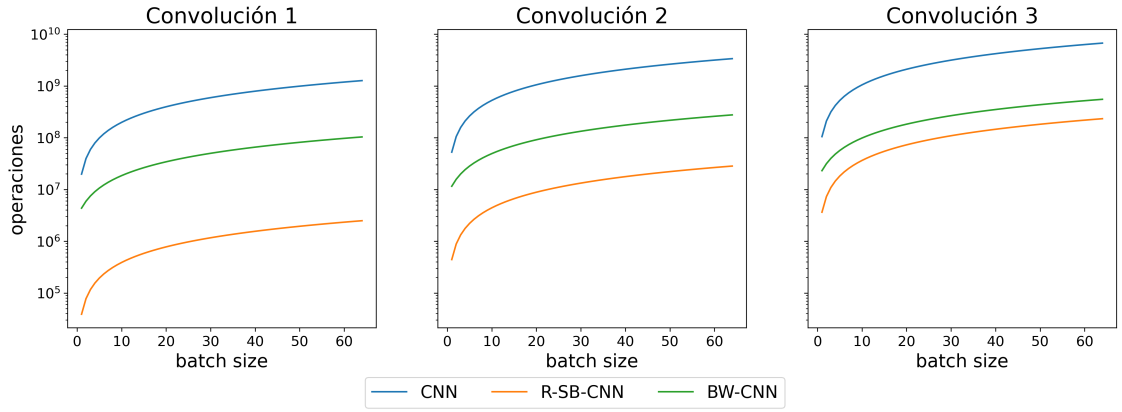


Figura 5.5: Número de operaciones de punto flotante realizadas en cada capa convolucional con respecto al tamaño del *batch*

presentan el problema de requerir un elevado número de parámetros en cada capa convolucional. En contraste, las arquitecturas BW-CNN requieren significativamente menos parámetros por capa, incluso presentan menos parámetros que la arquitectura CNN.

Tabla 5.14: Número de parametros por capa, utilizando un conjunto de datos con un tamaño de entrada de 128×128 para CNN, y BW-CNN, y una representación reducida de tamaño 16×16 para R-SB-CNN

Capa	CNN	R-SB-CNN	BW-CNN
Convolución 1	608	12 304	**96**
Convolución 2	6 432	131 136	**1 024**
Convolución 3	51 264	1 048 704	**8 192**
Total	58 304	1 192 144	**9 312**

La Tabla 5.15 muestra que el factor de reducción en el número de parámetros en las arquitecturas CNN y R-SB-CNN en comparación con BW-CNN mantiene valores constantes en todas las capas convolucionales. Esto se debe a que todas las arquitecturas en comparación mantienen un valor constante de parámetros por cada canal en un *kernel*. Mientras que R-SB-CNN tiene *kernels* de $2 \times 16 \times 16 + 2 = 514$ parámetros, para CNN se requieren de $5 \times 5 + 1 = 26$ parámetros, y para BW-CNN cada *kernel* solo requiere 4 parámetros (ver Sección 4.3.1). Esta marcada diferencia se ve reflejada en el factor de reducción de parámetros total, siendo de $\times$6,261 al comparar CNN con BW-CNN y de $\times$128,022 al comparar R-SB-CNN con BW-CNN.

El alto número de parámetros necesarios por la arquitectura R-SB-CNN puede ocasionar problemas durante el entrenamiento. En los experimentos realizados, se experimentaron interrupciones constantes del entrenamiento

Tabla 5.15: Reducción en parámetros de las arquitecturas BW-CNN en comparación con CNN y R-SB-CNN

Capa	CNN	R-SB-CNN
Convolución 1	×6,333	×128,167
Convolución 2	×6,281	×128,062
Convolución 3	×6,258	×128,016
Total	×6,261	×128,022

debido a problemas de memoria al calcular los gradientes de todos los parámetros de R-SB-CNN. Además, considerando que las arquitecturas usadas en estos experimentos no incluyen muchas capas convolucionales, se infiere que las arquitecturas R-SB-CNN no son adecuadas para trabajar con arquitecturas profundas, principalmente en dispositivos con poca memoria. Por el contrario, en el caso de CNN, su bajo número de parámetros por *kernel* permiten implementar arquitecturas profundas. Esta cualidad también se observa en las arquitecturas BW-CNN, además, considerando que las arquitecturas BW-CNN tienen una exactitud similar y requieren un menor número de operaciones en inferencia que CNN, se deduce que las arquitecturas BW-CNN presentan una opción viable para trabajar con imágenes grandes y arquitecturas profundas.

5.3. Resumen

La Tabla 5.16 muestra las características de las arquitecturas analizadas en esta sección: CNN, Redes de Fourier en el estado del arte (EF-CNN y SB-CNN), sus variantes tipo-R y LT, y BW-CNN. Note que las arquitecturas propuestas en esta investigación se encuentran subrayadas.

En cada arquitectura se describe el número de parámetros requeridos, las operaciones de punto flotante realizadas en inferencia y su exactitud al trabajar con imágenes pequeñas y grandes. Note que, en el caso de las arquitecturas tipo-LT, no se estudia el caso de imágenes grandes, esto se debe a que se presentaron inconvenientes en el entrenamiento debido al elevado consumo de memoria, lo cual contradice el objetivo de esta tesis, que es proponer arquitecturas de redes neuronales ligeras. Tampoco se incluye un análisis de las arquitecturas BW-CNN al trabajar con imágenes pequeñas. Esto se debe a que la técnica de filtros Butterworth requiere representaciones de entrada considerablemente grandes para caracterizar adecuadamente este tipo de filtros.

En el caso de las imágenes pequeñas, se observa que las Redes de Fourier en el estado del arte presentan un mayor número de parámetros y de operaciones de

Tabla 5.16: Características de las arquitecturas CNN, tipo-R, tipo-LT y BW-CNN

Arq.	Parámetros		Operaciones		Exactitud	
	I. pequeñas	I. grandes	I. pequeñas	I. grandes	I. pequeñas	I. grandes
CNN	Poco consumo de memoria	Poco consumo de memoria	Ineficiente	Ineficiente	Alta exactitud	Alta exactitud
Redes de Fourier	Alto consumo de memoria	No analizado	Más eficiente que CNN	No analizado	Ligeramente menor que CNN	No analizado
tipo-R	Ligeramente mayor a CNN, menor que Redes de Fourier	Alto consumo de memoria, menor que Redes de Fourier	Muy eficiente	Muy eficiente	La misma que Redes de Fourier	Alta exactitud, mayor a CNN en algunos casos, igual que Redes de Fourier
tipo-LT	Ligeramente mayor a tipo-R	No aplica	Ligeramente mayor a tipo-R	No aplica	Mayor a tipo-R, comparable con CNN en algunos casos	No aplica
BW-CNN	No aplica	Poco consumo de memoria, menor a CNN	No aplica	Eficiente	No aplica	Alta exactitud, comparable con CNN

punto flotantes en inferencia que sus variantes tipo-R. Además, si consideramos que las arquitecturas tipo-R presentan la misma exactitud que sus arquitecturas base, entonces resulta más conveniente optar por estas arquitecturas o incluso por arquitecturas tipo-LT, con un costo computacional adicional pero aún menor que las Redes de Fourier base. Al comparar con las arquitecturas CNN, se observa que las arquitecturas tipo-R y LT presentan una opción viable para reemplazar a CNN en escenarios donde se requiera reducir el número de operaciones en inferencia, aunque en algunos casos CNN alcanzan una precisión superior y requieren ligeramente menos parámetros en sus *kernels*.

En el caso de imágenes grandes, se observa que las arquitecturas tipo-R presentan exactitud superior a CNN, con una considerable reducción en las operaciones en inferencia. Sin embargo, el gran tamaño de sus *kernels* provoca un grave incremento en memoria, lo que dificulta tanto su entrenamiento como implementación, principalmente en arquitecturas profundas. Por otro lado, las arquitecturas BW-CNN presentan un menor número de parámetros que

la arquitecturas tipo-R e incluso menor que CNN. Además, realizan menos operaciones que CNN y mantienen una precisión similar.

En general, si se trabaja con imágenes grandes, tanto las arquitecturas tipo-R como BW-CNN son buenas opciones a considerar en lugar de CNN. Por un lado las arquitecturas tipo-R presentan una mejor exactitud, incluso mejor que CNN en algunos casos, pero involucran un alto consumo de memoria que podría dificultar la implementación de redes profundas. Por otro lado, las arquitecturas BW-CNN son una mejor opción en términos de uso de memoria y consiguen una precisión similar a CNN. La elección de cada arquitectura dependerá de las características del hardware disponible.

Capítulo 6

Conclusiones y trabajo futuro

En este trabajo se presentaron dos enfoques para el problema de clasificación de imágenes, que aprovechan las propiedades del teorema de la convolución para reducir el costo computacional de las CNN. Enseguida se describen los principales logros obtenidos y se emboza el trabajo futuro recomendando alrededor de esta investigación.

En el primer enfoque, aplicable en imágenes pequeñas, se plantearon las arquitecturas tipo-R y tipo-LT. Las arquitecturas tipo-R trabajan con una representación reducida, construida con un subconjunto de componentes frecuenciales obtenidos de las imágenes. Se observó experimentalmente que las arquitecturas tipo-R son equivalentes a las Redes de Fourier disponibles el estado del arte que utilizan capas *Spectral Pooling*. Además, las arquitecturas tipo-R presentan una notable reducción tanto en el número de parámetros como en las operaciones de punto flotante ejecutadas durante la inferencia, en comparación con las Redes de Fourier. Sin embargo, las arquitecturas tipo-R presentan pérdida de información. Este inconveniente es resuelto con las arquitecturas tipo-LT, las cuales, junto con una nueva capa denominada *Linear Transform*, crean de forma dinámica una representación reducida que conserva la información más relevante de la representación frecuencial de las imágenes. Dicha capa requiere de un ligero incremento en el número de parámetros y operaciones de punto flotante, en comparación con arquitecturas tipo-R. Experimentalmente se observó que la representación reducida construida por la capa *Linear Transform* ofrece una mejor representación de la información, favoreciendo la exactitud. Al comparar las arquitecturas propuestas con CNN, se observó que tanto las arquitecturas tipo-R como LT presentan una significativa reducción de operaciones de punto flotante, además de conseguir un número de parámetros similares a CNN, al utilizarse filtros 5×5. Más aún, se observó que las arquitecturas tipo-LT presentan una precisión similar a CNN, en las bases de datos MNIST y Fashion-MNIST. Estas ventajas destacan en comparación con las Redes de Fourier encontradas en el estado del arte, lo que posiciona a las arquitecturas propuestas de manera más favorable para reemplazar a las redes CNN al trabajar con imágenes pequeñas.

Sin embargo, al trabajar con imágenes grandes, se observa que las arquitecturas tipo-R presentan un incremento en el número de parámetros. Para afrontar este inconveniente se propusieron las arquitecturas BW-CNN, estrategia basada en una función generadora de filtros Butterworth. Ésta utiliza un bajo número de parámetros y una nueva capa de *pooling*, denominada *Spectral Average Pooling*, que aplica la transformación *Average Pooling* por separado a la parte real e imaginaria de la representación frecuencial. Se observó que, al trabajar con imágenes grandes, BW-CNN logra una notable reducción en el número de parámetros, en comparación con las arquitecturas tipo-R y un menor costo de memoria en comparación con las CNN. Además, BW-CNN realiza considerablemente menos operaciones de punto flotante en inferencia que CNN, principalmente al incrementarse el tamaño del *batch*. Sin embargo, en términos de exactitud, las arquitecturas tipo-R obtienen los mejores resultados, mientras que las arquitecturas BW-CNN y CNN mantienen entre sí niveles de exactitud similares. En general, al trabajar con imágenes grandes, las arquitecturas BW-CNN muestran una buena relación entre exactitud y costo computacional. Por el contrario, las arquitecturas tipo-R presentan una exactitud superior a CNN en algunos casos, pero su alto costo en memoria dificulta su implementación en dispositivos con recursos limitados y en arquitecturas profundas.

Las características de las arquitecturas propuestas son ideales para implementarse en dispositivos con recursos limitados. Ya que el enfoque propuesto presenta un desempeño similar a CNN, pero con un menor costo computacional. Además, las operaciones involucradas en el trabajo propuesto, incluyendo el algoritmo FFT, el producto de Hadamard y las capas propuestas, son paralelizables, por lo cual se pueden plantear diversas estrategias para agilizar la inferencia de este tipo de arquitecturas.

6.1. Trabajo futuro

Como trabajo futuro, se sugiere propone nuevas alternativas para funciones de activación y capas de *pooling* para Redes de Fourier, considerando en ambos casos alternativas con bajo costo computacional y adecuadas para trabajar con la representación frecuencial de las imágenes.

Durante el desarrollo de esta investigación se identificaron ciertos aspectos en las Redes de Fourier que aún no han sido investigados y que podrían mejorar su precisión. En primer lugar, se observó que la representación del color en el espacio de Fourier permite mejorar los niveles de exactitud, al compararse con representaciones en niveles de gris, principalmente en imágenes naturales como

es el caso de CIFAR-10. Por consiguiente, se sugiere como trabajo futuro proponer nuevas representaciones de la información de imágenes a color en el dominio de Fourier.

Por otro lado, se detectó que las Redes de Fourier, al igual que las arquitecturas CNN, presentan inconvenientes al trabajar con conjuntos de datos con clases desbalanceadas. Por lo cual, es imprescindible estudiar la aplicación de técnicas de aumento de datos. Cabe hacer notar que la mayoría de técnicas basadas en transformaciones geométricas no son adecuadas para utilizarse en el dominio de Fourier. Esto se debe a que la transformada de Fourier es invariante bajo diversas transformaciones geométricas, como el rescalamiento, traslación y rotación. Por esta razón, es necesario explorar nuevas alternativas para realizar aumento de datos en el dominio de Fourier sin involucrar técnicas geométricas.

Por último, se propone expandir el análisis de las Redes de Fourier a otras tareas de visión por computadora, como lo son la detección o segmentación de imágenes, contextos en donde sería valioso mantener información espacial y frecuencial.

Bibliografía

[1] Sayed Omid Ayat, Mohamed Khalil-Hani, Ab Al-Hadi Ab Rahman, and Hamdan Abdellatef. Spectral-based convolutional neural network without multiple spatial-frequency domain switchings. *Neurocomputing*, 364:152–167, 2019.

[2] K Simonyan and A Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*, pages 1–14, 2015.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[4] Duo Li, Jie Hu, Changhu Wang, Xiangtai Li, Qi She, Lei Zhu, Tong Zhang, and Qifeng Chen. Involution: Inverting the inherence of convolution for visual recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12321–12330, 2021.

[5] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[6] Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in neural information processing systems*, 34:24261–24272, 2021.

[7] Oren Rippel, Jasper Snoek, and Ryan P Adams. Spectral representations for convolutional neural networks. *Advances in neural information processing systems*, 28, 2015.

[8] Harry Pratt, Bryan Williams, Frans Coenen, and Yalin Zheng. Fcnn: Fourier convolutional neural networks. In *Machine Learning and Knowledge Discovery*

in Databases: European Conference, ECML PKDD 2017, Skopje, Macedonia, September 18–22, 2017, Proceedings, Part I 17, pages 786–798. Springer, 2017.

[9] C Trabelsi, O Bilaniuk, Y Zhang, D Serdyuk, S Subramanian, JF Santos, S Mehri, N Rostamzadeh, Y Bengio, and CJ Pal. Deep complex networks int. In *Conf. on Learning Representations*, 2018.

[10] Thomio Watanabe and Denis F Wolf. Image classification in frequency domain with 2srelu: a second harmonics superposition activation function. *Applied Soft Computing*, 112:107851, 2021.

[11] Yuna Han and Byung-Woo Hong. Deep learning based on fourier convolutional neural network incorporating random kernels. *Electronics*, 10(16):2004, 2021.

[12] Jakub Zak, Anna Korzynska, Antonina Pater, and Lukasz Roszkowiak. Fourier transform layer: A proof of work in different training scenarios. *Applied Soft Computing*, 145:110607, 2023.

[13] John G Proakis. *Digital signal processing: principles, algorithms, and applications, 4/E*. Pearson Education India, 2007.

[14] James W Cooley and John W Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of computation*, 19(90):297–301, 1965.

[15] Trabelsi Chiheb, O Bilaniuk, D Serdyuk, et al. Deep complex networks. In *International Conference on Learning Representations*, 2017.

[16] Martin Arjovsky, Amar Shah, and Yoshua Bengio. Unitary evolution recurrent neural networks. In *International conference on machine learning*, pages 1120–1128. PMLR, 2016.

[17] Nitzan Guberman. On complex valued convolutional neural networks. *arXiv preprint arXiv:1602.09046*, 2016.

[18] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[19] David H Hubel and Torsten N Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106, 1962.

[20] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

[21] Nipun Sadvilkar Mirza Rahim Baig, Thomas V. Joseph. *The Deep Learning Workshop*. Packt, 2020.

[22] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[23] Chris Solomon and Toby Breckon. *Fundamentals of Digital Image Processing: A practical approach with examples in Matlab*. John Wiley & Sons, 2011.

[24] Rafael C Gonzales and Paul Wintz. *Digital image processing*. Addison-Wesley Longman Publishing Co., Inc., 1987.

[25] Alex Smola and SVN Vishwanathan. Introduction to machine learning. *Cambridge University, UK*, 32(34):2008, 2008.

[26] David L Olson and Dursun Delen. *Advanced data mining techniques*. Springer Science & Business Media, 2008.

[27] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[28] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. Dynamic filter networks. *Advances in neural information processing systems*, 29, 2016.

[29] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[30] Julio Zamora Esquivel, Adan Cruz Vargas, Paulo Lopez Meyer, and Omesh Tickoo. Adaptive convolutional kernels. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[31] Michael Mathieu, Mikael Henaff, and Yann LeCun. Fast training of convolutional networks through ffts. *arXiv preprint arXiv:1312.5851*, 2013.

[32] Jinhua Lin, Lin Ma, and Yu Yao. A fourier domain acceleration framework for convolutional neural networks. *Neurocomputing*, 364:254–268, 2019.

[33] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[34] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[35] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. *Master thesis, University of Toronto*, 2009.

[36] Jakob Nikolas Kather, Frank Gerrit Zöllner, Francesco Bianconi, Susanne M Melchers, Lothar R Schad, Timo Gaiser, Alexander Marx, and Cleo-Aron Weis. Collection of textures in colorectal cancer histology. *Zenodo https://doi.org/10*, 5281, 2016.

[37] Philipp Tschandl, Cliff Rosendahl, and Harald Kittler. The ham10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. *Scientific data*, 5(1):1–9, 2018.

[38] Karar Ali, Zaffar Ahmed Shaikh, Abdullah Ayub Khan, and Asif Ali Laghari. Multiclass skin cancer classification using efficientnets–a first step towards preventing skin cancer. *Neuroscience Informatics*, 2(4):100034, 2022.

[39] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[40] Ardan Adi Nugroho, Isnandar Slamet, and Sugiyanto Sugiyanto. Skins cancer identification system of hamloooo skin cancer dataset using convolutional neural network. In *AIP conference proceedings*, volume 2202. AIP Publishing, 2019.

[41] Saksham Bassi and Atharva Gomekar. Deep learning diagnosis of pigmented skin lesions. In *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pages 1–6. IEEE, 2019.

[42] Dorin Moldovan. Transfer learning based method for two-step skin cancer images classification. In *2019 E-Health and Bioengineering Conference (EHB)*, pages 1–4. IEEE, 2019.

[43] Emrah ÇEVİK and Kenan ZENGİN. Classification of skin lesions in dermatoscopic images with deep convolution network. *Avrupa Bilim ve Teknoloji Dergisi*, pages 309–318, 2019.