Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Computación

Preprocesamiento De Imágenes Para La Recuperación De Fase En Holografía Cuántica

Tesis para obtener el título de:

Maestra en Ciencias de la Computación

Presenta:
María del Rocío Hernández Flores

Directores de tesis:

Dra. Barbara Emma Sánchez Rinza

Dr. Mario Rossainz López

Enero 2026

Resumen

En este trabajo se desarrolla y evalúa un sistema híbrido de procesamiento y aprendizaje automático para la reconstrucción de patrones de interferencia y la recuperación indirecta de información de fase en el contexto de holografía cuántica con luz no detectada. El proceso parte de un conjunto de imágenes obtenidas mediante técnicas de phase shifting, a partir de las cuales se busca generar reconstrucciones de mayor fidelidad y evaluar su utilidad para el cálculo de fase por el método de cuatro pasos.

El primer componente del sistema utiliza técnicas de representación dispersa, combinando Dictionary Learning y Matching Pursuit (MP) para aprender diccionarios adaptados a los patrones presentes en las imágenes. Se emplearon estrategias de optimización basadas en aprendizaje evolutivo para seleccionar los hiperparámetros del proceso, incluyendo tamaño del diccionario, número de átomos, ventana de Hann y tamaño de parche. Tras el entrenamiento del diccionario, las imágenes fueron reconstruidas mediante Orthogonal Matching Pursuit (OMP), lo que permitió obtener versiones más estables y estructuralmente consistentes de los patrones originales. Estas reconstrucciones fueron evaluadas mediante métricas cuantitativas como PSNR, SSIM, RMSE y correlación.

En una segunda etapa se entrenó un autoencoder convolucional diseñado para generar de manera directa las tres imágenes desplazadas a partir de una sola imagen de referencia. Se evaluaron diferentes tamaños de conjunto de entrenamiento (1000 y 10000 parches) y se analizaron las reconstrucciones obtenidas. A pesar de que el modelo mostró capacidad para capturar la estructura global de los patrones, las imágenes generadas no alcanzaron la nitidez requerida para un cálculo confiable de la fase, presentando artefactos visuales y pérdida de resolución.

Finalmente, se realizó el cálculo de la fase tanto para las imágenes originales como para las reconstruidas mediante MP. Las fases recuperadas a partir del MP mostraron estructuras más regulares y menos ruido respecto a las obtenidas del autoencoder, por lo que este último no fue considerado adecuado para el análisis de fase en su estado actual. Las métricas promedio obtenidas entre imágenes reconstruidas fueron: PSNR = 10.33, SSIM = 0.015, RMSE = 1.9129 y correlación = 0.0852, valores que reflejan las limitaciones inherentes al tamaño de parche, la resolución de las imágenes y la presencia de ruido en los datos experimentales.

Este trabajo demuestra que la combinación de representación dispersa y reconstrucción por MP constituye una herramienta viable para mejorar la calidad estructural de los patrones interferométricos en holografía cuántica con luz no detectada. Sin embargo, también evidencia la necesidad de explorar modelos generativos más profundos o arquitecturas especializadas que permitan mejoras sustanciales en la calidad y resolución de las imágenes generadas, especialmente para aplicaciones donde la fase es el parámetro fundamental.

Abstract

This work develops and evaluates a hybrid system that combines signal processing and machine learning for the reconstruction of interference patterns and the indirect recovery of phase information in the context of quantum holography with undetected light. The study begins with a dataset of phase-shifting images, from which the goal is to generate higher-fidelity reconstructions and assess their suitability for phase estimation using the four-step method.

The first component of the system employs sparse representation techniques, combining Dictionary Learning and Matching Pursuit (MP) to learn dictionaries adapted to the structures present in the images. An evolutionary optimization scheme was used to select hyperparameters such as dictionary size, number of atoms, Hann windowing, and patch size. After dictionary training, image reconstruction was performed using Orthogonal Matching Pursuit (OMP), producing structurally consistent and more stable versions of the original patterns. These reconstructions were quantitatively evaluated using PSNR, SSIM, RMSE, and correlation metrics.

In the second stage, a convolutional autoencoder was trained to directly generate the three phase-shifted images from a single reference image. Different training set sizes (1000 and 10,000 patches) were tested, and the resulting predictions were analyzed. Although the model demonstrated the ability to capture the global structure of the patterns, the generated images lacked the resolution required for reliable phase estimation, exhibiting grid artifacts and loss of fine detail.

Finally, phase was computed for both the original images and those reconstructed through MP. The phases recovered from the MP reconstructions displayed smoother structures and reduced noise compared to those produced by the autoencoder, which was therefore not considered adequate for phase analysis in its current form. The average metrics obtained between reconstructed images were: PSNR = 10.33, SSIM = 0.015, RMSE = 1.9129, and correlation = 0.0852. These values reflect the limitations imposed by patch size, image resolution, and inherent noise in the experimental data.

Overall, this work shows that the combination of sparse representation and MP-based reconstruction is a viable approach for enhancing the structural quality of interferometric patterns in quantum holography with undetected light. However, it also highlights the need to explore deeper generative models or specialized architectures capable of producing higher-resolution and more accurate reconstructions, especially in applications where phase is the key parameter.

Agradecimientos

Agradezco profundamente al Comité de Posgrado y a la Secretaría de Ciencia, Humanidades, Tecnología e Innovación (SECIHTI) por el apoyo brindado durante el desarrollo de mis estudios de maestría. Su respaldo académico y económico hizo posible la realización de este trabajo.

El personal autor agradece al Laboratorio Nacional de Supercómputo del Sur-este de México (LNS), perteneciente al padrón de laboratorios nacionales CONA-HCYT, por los recursos computacionales, el apoyo y la asistencia técnica brindados, a través del proyecto No. 202501027C.

Expreso mi sincero agradecimiento a mis asesores, Dra. Barbara Sánchez Rinza y Dr. Dr. Mario Rossainz López, cuya guía, paciencia y conocimientos fueron fundamentales para la realización de esta investigación. Sus observaciones y orientación aportaron claridad en cada etapa del proyecto.

Extiendo también mi gratitud a los profesores y profesoras de la Facultad, en especial a la Dra. Meliza Contreras González, al Dr. J. Arturo Olvera López, y al Dr. Iván Olmos Pineda , quienes contribuyeron a mi formación académica y ofrecieron retroalimentación valiosa.

Agradezco y reconozco la colaboración y apoyo de mis compañeros y compañeras de posgrado, cuya disposición para discutir ideas, compartir experiencias y ofrecer ánimos fue una parte esencial durante este proceso.

Contenido

Lista de Figuras

Lista de Tablas

Lista de Algoritmos

CAPÍTULO 1

INTRODUCCIÓN

1.1. Introducción A La Holografía

1.1.1. Holografía Clásica

La holografía es un método de producción de imágenes tridimensionales realizado en dos etapas: registro y reconstrucción.

En la fase de registro, un haz de referencia se dirige directamente hacia un medio de grabación (como una placa fotográfica), mientras que el haz objeto ilumina el objeto y luego se refleja o dispersa hacia el mismo medio de grabación; la superposición de ambos haces en la placa crea un patrón de interferencia que contiene información sobre la amplitud y la fase de la luz reflejada por el objeto. Para reconstruir la imagen, se ilumina el holograma con un haz de luz similar al de referencia lo que genera una imagen tridimensional del objeto original, que puede ser observada desde diferentes ángulos, dando la ilusión de profundidad (Figura 1.1).

La holografía clásica captura tanto la amplitud como la fase de la luz, lo que le permite reconstruir fielmente el objeto en tres dimensiones [1].

Entre sus aplicaciones se pueden mencionar la creación de imágenes tridimensionales que pueden ser vistas sin necesidad de gafas especiales, la medición precisa de formas y deformaciones en superficies y en cuestiones de seguridad como el uso de hologramas utilizados en tarjetas de crédito, pasaportes y billetes para evitar falsificaciones.

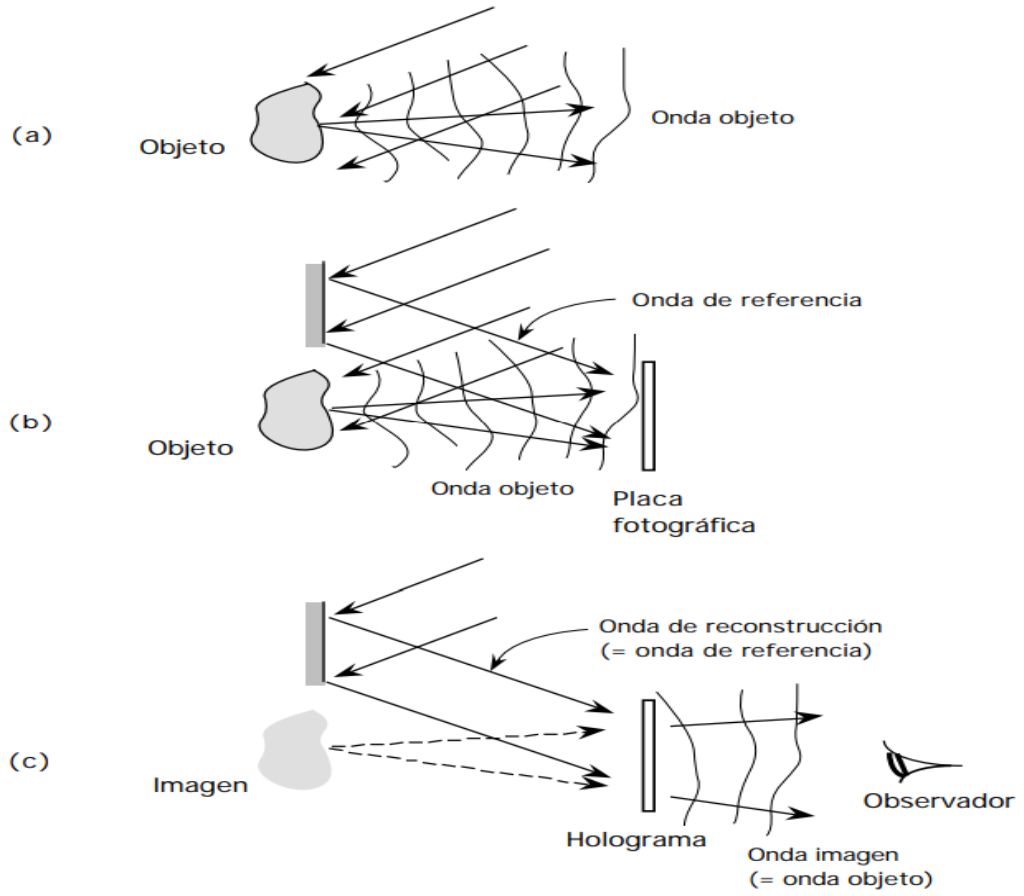


Figura 1.1 (a) Onda proveniente de un objeto (b) Esquema de registro de un holograma (c) Reconstrucción de un holograma [1].

1.1.2. Holografía Cuántica

La holografía cuántica utiliza principios de la mecánica cuántica, especialmente el entrelazamiento cuántico y la superposición, para grabar y reconstruir hologramas.

En algunas configuraciones de holografía cuántica (haces de luz no detectados), se utilizan fotones entrelazados en lugar de luz láser coherente clásica. Un fotón (el fotón "objeto") interactúa con el objeto, mientras que su par entrelazado (el fotón "de referencia") no lo hace. A diferencia de la holografía tradicional, esta metodología permite generar imágenes sin la necesidad de que los fotones que interactúan directamente con el objeto sean detectados, permitiendo el desarrollo de nuevas técnicas de análisis en áreas donde el objeto puede ser frágil o sensible a la luz.

El entrelazamiento cuántico es un fenómeno en el que dos partículas, como los fotones, quedan correlacionadas de tal manera que el estado de una determina instantáneamente el estado de la otra, independientemente de la distancia entre ellas [2]. Esta propiedad es esencial en la holografía cuántica, ya que se emplea para transmitir información entre los fotones entrelazados. La generación de fotones entrelazados se logra a través de un proceso llamado "baja conversión paramétrica espontánea" (SPDC, por sus siglas en inglés), donde un fotón de alta energía interactúa con un cristal no lineal, dividiéndose en dos fotones de menor energía que conservan una correlación cuántica [3].

En el esquema de la holografía cuántica, un par de fotones entrelazados se divide en dos caminos. Uno de los fotones, denominado "fotón testigo", interactúa con el objeto que se desea analizar, mientras que el otro fotón, llamado "fotón referencia", no lo hace. La información cuántica del fotón testigo se transfiere al fotón referencia debido al entrelazamiento, lo que permite que al medir el fotón referencia, se pueda recuperar la información sobre el objeto sin necesidad de detectar al fotón testigo [4]. Esta propiedad se ha denominado "holografía con luz no detectada", ya que la imagen se reconstruye a partir de la información del fotón no interactuante (Figura 1.2).

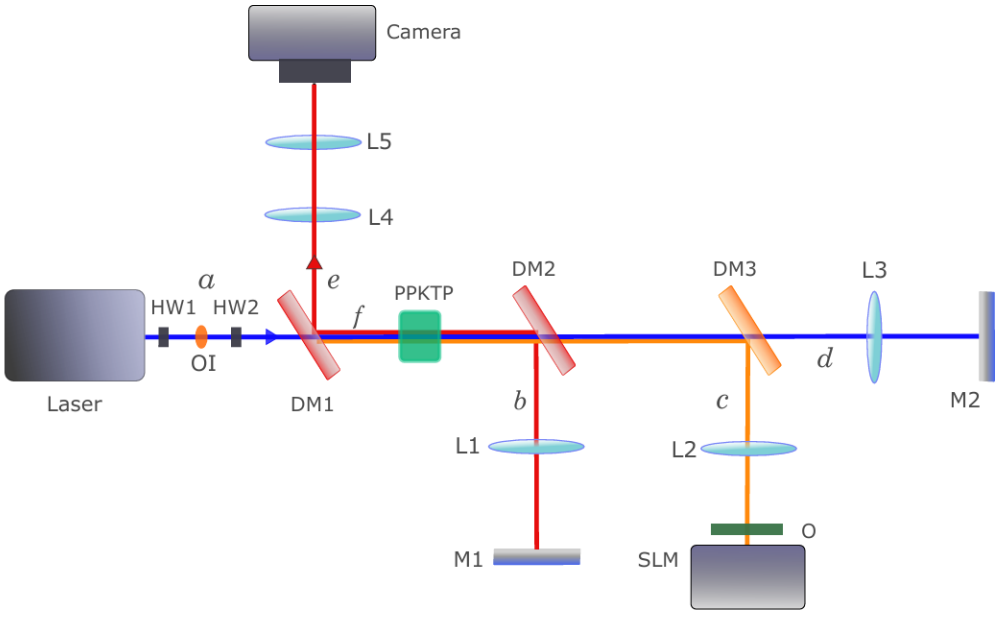


Figura 1.2 Ejemplo de arreglo experimental de holografía cuántica. Abreviaturas: HW1–HW2: half-wave plates (placas de media onda); OI: optical isolator (aislador óptico); DM1–DM3: dichroic mirrors (espejos dicroicos); L1–L5: lenses (lentes); PPKTP: periodically poled KTiOPO$_4$ (cristal no lineal); SLM: spatial light modulator (modulador

espacial de luz); M1–M2: mirrors (espejos); O: objeto. Las letras a, b, c, d, e y f representan caminos ópticos o distancias específicas dentro del sistema.

Esta técnica presenta varias ventajas ya que, al no requerir la detección directa del fotón que interactúa con el objeto, se minimiza el riesgo de daño a muestras sensibles, como tejidos biológicos o materiales delicados [5]. Además, el uso del entrelazamiento cuántico permite superar algunas limitaciones clásicas de la óptica, como el ruido en la imagen y las restricciones en la resolución espacial, abriendo el camino a nuevas aplicaciones en áreas como la microscopía cuántica y la imagenología médica [6].

En otras configuraciones se utiliza la no localidad cuántica en la que la información sobre el objeto se graba sin que el fotón de referencia interactúe directamente con él, aprovechando la correlación cuántica entre los pares de fotones.

Ambas formas de holografía representan avances significativos en la captura y reconstrucción de imágenes, pero desde fundamentos físicos distintos y con aplicaciones que pueden complementarse en diversas áreas científicas y tecnológicas.

1.1.3. Holografía Digital

La holografía digital es una técnica avanzada de imagen que permite la reconstrucción tridimensional de objetos a partir de patrones de interferencia registrados digitalmente.

La holografía digital se fundamenta en la captura y el procesamiento de la interferencia entre un haz de referencia y un haz objeto. Un haz láser coherente se divide en dos: uno ilumina el objeto y se dispersa antes de ser registrado, mientras que el otro actúa como referencia. La superposición de ambos en un detector digital, como un sensor CCD o CMOS, genera un holograma que contiene información de amplitud y fase de la luz reflejada por el objeto [7].

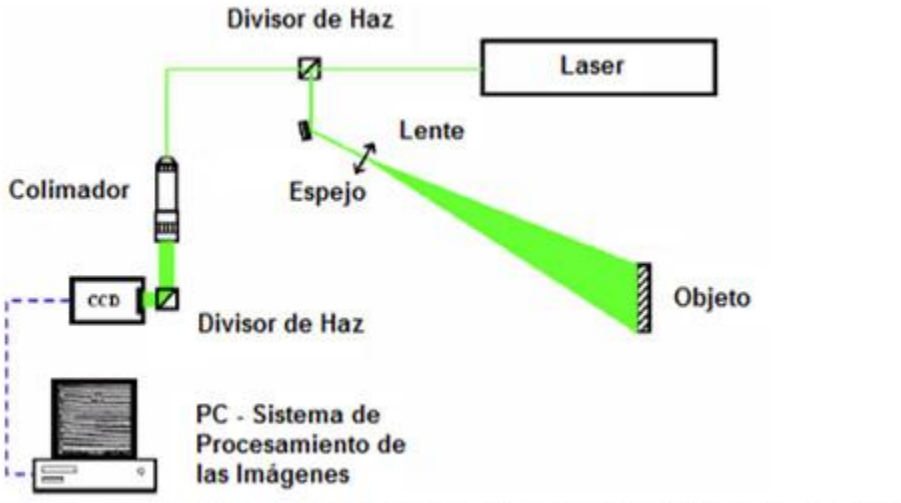


Figura 1.3 Ejemplo de arreglo físico para Holografía Digital [100]

A diferencia de la holografía tradicional, la reconstrucción de la imagen en la holografía digital se realiza mediante algoritmos numéricos en lugar de técnicas ópticas lo que permite la manipulación de los datos holográficos, la mejora de la calidad de la imagen y la implementación de técnicas computacionales avanzadas como la reconstrucción tomográfica y el filtrado de ruido [8].

Existen diferentes configuraciones para la captura de hologramas digitales, entre ellas:

1. Holografía Digital en Transmisión: La luz atraviesa el objeto antes de interferir con el haz de referencia y es útil en aplicaciones biomédicas y en la caracterización de materiales transparentes [9].
2. Holografía Digital en Reflexión: Se registra la luz reflejada por el objeto, lo que permite obtener información sobre su estructura superficial y es usada comúnmente en metrología y pruebas no destructivas [10].
3. Holografía Digital de Frecuencia Espacial: Emplea múltiples longitudes de onda o ángulos de iluminación para mejorar la calidad de reconstrucción y la resolución de la imagen holográfica [11].

La reconstrucción digital del holograma se basa en la transformada de Fourier y otros métodos computacionales. Técnicas como la propagación angular y la propagación de difracción de Fresnel permiten extraer la información de fase, lo que posibilita la reconstrucción tridimensional del objeto [12].

La holografía digital ha demostrado ser una herramienta valiosa en diversas áreas científicas y tecnológicas como en biomedicina donde permite la visualización de estructuras celulares y tisulares en 3D sin necesidad de tinción [13]; en la metrología e ingeniería, aplicada a la inspección de materiales y componentes estructurales mediante técnicas de interferometría holográfica [14]; en la seguridad y autenticación, empleada en la generación de hologramas de seguridad para la protección de documentos y productos comerciales [15], y en la microscopía holográfica digital, donde permite el análisis cuantitativo de muestras biológicas con alta resolución y sin contacto físico [16].

1.2. Generación de Hologramas

Para recuperar la fase, en muchos casos se necesitan múltiples mapas de intensidad, como en la holografía con desplazamiento de fase y la proyección axial de múltiples intensidades alternas. Dada su excelente capacidad de mapeo, la red neuronal puede utilizarse para generar otros hologramas relevantes a partir de hologramas conocidos, lo que permite la recuperación de fase que requiere múltiples hologramas. En este enfoque, la entrada y la salida generalmente pertenecen a la misma modalidad de imagen con alta similitud de características, por lo que es más fácil para la red neuronal aprender. Además, el conjunto de datos se recolecta solo a partir de registros experimentales o generación mediante simulación, sin la necesidad de recuperar la fase como referencia mediante métodos convencionales.

Zhang et al. [17,18] propusieron por primera vez la idea de generar hologramas con hologramas antes de la recuperación de fase mediante el método convencional. A partir de un único holograma, los otros tres hologramas con desplazamientos de fase de $\pi/2$, π y $\frac{3\pi}{2}$ fueron generados simultáneamente por la Y-Net [19], y luego se implementó la recuperación de fase mediante el método de desplazamiento de fase en cuatro pasos.

La motivación para inferir hologramas en lugar de la fase a través de una red es que, para diferentes tipos de muestras, las diferencias espaciales entre sus hologramas son significativamente menores que las de su fase. En consecuencia, esta recuperación de fase basada en la generación de hologramas tiene mejor capacidad de generalización que la recuperación de fase a partir de

hologramas directamente con la red neuronal, especialmente cuando las diferencias en las características espaciales de la fase entre los conjuntos de datos de entrenamiento y prueba son relativamente grandes [18]. Dado que el desplazamiento de fase entre los hologramas generados es igual, Yan et al. [20] propusieron generar hologramas con desplazamiento de fase sin ruido utilizando una red generativa antagónica (GAN) simple de extremo a extremo en un modo de concatenación secuencial.

Posteriormente, para un mejor rendimiento en el equilibrio entre los detalles espaciales y la información semántica de alto nivel, Zhao et al. [21] aplicaron la red progresiva de restauración de imágenes en múltiples etapas (MPRNet) [22] para la generación de hologramas con desplazamiento de fase. Huang et al. [23] y Wu et al. [24] luego ampliaron este enfoque de métodos de desplazamiento de fase de cuatro pasos a tres pasos y dos pasos, respectivamente.

Luo et al. [25] propusieron generar hologramas con diferentes distancias de desenfoque a partir de un holograma mediante una red neuronal, y luego lograr la recuperación de fase con proyección alterna. Similar al trabajo de Zhang et al. [18], demostraron que el uso de redes neuronales con menor diferencia entre el dominio fuente y el dominio objetivo podría mejorar la capacidad de generalización.

En cuanto a la holografía de múltiples longitudes de onda, Li et al. [26,27] aprovecharon una red neuronal para generar un holograma de otra longitud de onda a partir de uno o dos hologramas de longitud de onda conocida, logrando así holografía de dos y tres longitudes de onda. Al mismo tiempo, Xu et al. [28] realizaron una holografía de dos y tres longitudes de onda en una sola toma generando los hologramas correspondientes de longitud de onda única a partir de un holograma de dos o tres longitudes de onda con interferencia de información.

1.3 Técnicas de Recuperación de Fase

La luz, como onda electromagnética, tiene dos componentes esenciales: amplitud y fase [26]. Los detectores ópticos miden la intensidad que es proporcional al cuadrado de la amplitud del campo de luz. Sin embargo, no pueden capturar la fase del campo de luz debido a su limitada frecuencia de muestreo [30].

En muchos escenarios de aplicación, la fase, en lugar de la amplitud del campo de luz, es la que lleva la información principal de las muestras [31-34].

A medida que el campo de luz se propaga, el retardo de fase también provoca cambios en la distribución de la amplitud; por lo tanto, podemos registrar la amplitud del campo de luz propagado y posteriormente calcular la fase correspondiente. Esta operación generalmente recibe diferentes nombres según el dominio de la aplicación; por ejemplo, se denomina imagen de fase cuantitativa (quantitative phase imaging, QPI) en biomedicina, imagen de difracción coherente (coherent diffraction imaging, CDI), que es el término más comúnmente utilizado en óptica de rayos X y análogos no ópticos, como electrones y otras partículas, y censado de frente de onda en óptica adaptativa (adaptive optics, AO) [33] en astronomía y comunicaciones ópticas. En este caso, la forma de calcular la fase de un campo de luz a partir de sus mediciones de intensidad se le denomina recuperación de fase (phase recovery, PR).

Calcular la fase directamente a partir de una medición de intensidad después de la propagación suele estar mal condicionado. Suponiendo que se conoce el campo complejo en el plano del sensor, se puede calcular directamente el campo complejo en el plano de la muestra utilizando la propagación numérica. Sin embargo, el sensor solo registra la intensidad, pero pierde la fase y, además, se muestrea en píxeles de tamaño de área finita. Debido a esto, la distribución del campo complejo en el plano de la muestra generalmente no puede ser calculado de manera directa.

La recuperación de fase puede transformase en un problema bien condicionado o determinístico introduciendo información adicional, como en holografía o interferometría al introducir una onda de referencia [1,2], la detección de frente de onda de Shack-Hartmann que introduce una matriz de microlentes en el plano conjugado, y la ecuación de transporte de intensidad que requiere múltiples amplitudes a través del enfoque [35,36].

Este problema mal condicionado de recuperación de fase puede ser resuelto de manera iterativa mediante optimización, es decir, la recuperación de fase por medio del algoritmo de Gerchberg-Saxton-Fienup [37-39], el algoritmo de múltiples alturas [40-42], la ptychografía en espacio real [43-45] y la ptychografía en el espacio de Fourier [46, 47].

Si no se desea introducir información adicional, calcular la fase directamente a partir de una medición de intensidad propagada es un problema mal condicionado. Esta dificultad puede

superarse al incorporar conocimiento previo, también conocido como regularización. En el algoritmo de Gerchberg-Saxton (GS) [37], la intensidad en el plano de la muestra y el plano del sensor en el campo lejano registrados por el sensor se utilizan como restricciones. Un campo complejo se proyecta hacia adelante y hacia atrás entre estos dos planos usando la transformada de Fourier y se restringe iterativamente por la intensidad; de esta forma, el campo complejo resultante se acercará gradualmente a una solución. Fienup cambió la restricción de intensidad en el plano de la muestra por la restricción del diafragma (región de soporte), de modo que el sensor solo necesita registrar un mapa de intensidad, lo que dio lugar a los algoritmos de reducción de errores (ER) y, de entrada-salida híbrida (HIO) [38, 39]. Además de la restricción del diafragma, se pueden introducir otras restricciones físicas, como el histograma [48], la atomicidad [49] y la absorción [50], para reducir el mal condicionamiento de la recuperación de fase. Además, muchos tipos de priors de esparcimiento, como el dominio espacial [51], el dominio de gradiente [52, 53] y el dominio de wavelet [54], son regularizadores efectivos para la recuperación de fase.

Naturalmente, si el sensor registra más mapas de intensidad, habrá más conocimiento previo para la regularización, lo que reducirá aún más el mal condicionamiento del problema. Al mover el sensor axialmente, se registran los mapas de intensidad de diferentes distancias de desenfoque como una restricción de intensidad, y luego el campo complejo se calcula iterativamente como en el algoritmo GS, lo que se conoce como recuperación de fase de múltiples alturas [40 -42]. En este método de proyección alterna de múltiples intensidades axiales, la distancia entre el plano de la muestra y el plano del sensor generalmente se mantiene lo más cercana posible, de modo que se utiliza la propagación numérica para la proyección en lugar de la transformada de Fourier. Mientras tanto, con una posición fija del sensor, también se pueden registrar múltiples mapas de intensidad moviendo radialmente el diafragma cerca de la muestra, y luego se recupera el campo complejo iterativamente como en los algoritmos ER y HIO, lo que se conoce como ptychografía en el espacio real [51 - 53]. En este método de proyección alterna de múltiples intensidades radiales, cada restricción del diafragma adyacente se superpone con la otra y expande el campo de visión en el espacio real. Además, también es posible la proyección alterna de múltiples intensidades angulares. Al cambiar la restricción del diafragma del dominio espacial al dominio de la frecuencia con un sistema de lentes, se registran múltiples mapas de intensidad con diferente información de frecuencia cambiando el ángulo de la luz incidente, lo que se conoce como ptychografía de Fourier [46, 47]. Debido al cambio de ángulo de iluminación, se registra información de alta frecuencia

que originalmente excede la apertura numérica, expandiendo el ancho de banda de Fourier en el espacio recíproco.

Existen dos métodos de optimización no convexos más representativos, a saber, los algoritmos de flujo de Wirtinger [55] y flujo de amplitud truncado [56]. Estos pueden transformarse en problemas de optimización convexa a través de la programación semidefinida, como el algoritmo PhaseLift [57].

CAPÍTULO 2

PREPROCESAMIENTO DE IMÁGENES PARA GENERACIÓN DE HOLOGRAMAS

Objetivos Generales y Específicos del Proyecto

Objetivo General:

- Desarrollar un sistema que combine el algoritmo Matching Pursuit (MP) para el preprocesamiento y eliminación de ruido en imágenes holográficas y un modelo generativo para la síntesis de hologramas con desplazamientos de fase, optimizando los hiperparámetros del proceso mediante un algoritmo genético para mejorar la recuperación de información de fase y reducir la necesidad de capturar múltiples imágenes en experimentos holográficos.

Objetivos Específicos:

- Implementar el preprocesamiento de imágenes con Matching Pursuit.
- Optimizar los hiperparámetros del algoritmo Matching Pursuit.
- Desarrollar y entrenar un modelo generativo para la síntesis de hologramas desplazados.
- Validar la efectividad del sistema en la recuperación de fase.

Antecedentes

La holografía es una técnica que permite registrar la intensidad y reconstruir tanto la amplitud como la fase de una onda luminosa, lo que ha sido fundamental en el desarrollo de diversas aplicaciones ópticas, como la visualización en tres dimensiones, la metrología precisa y el análisis de micro y nanoestructuras [58]. La holografía digital ha ganado relevancia al utilizar sensores electrónicos, como cámaras, para registrar patrones de interferencia entre un haz de referencia y un haz que contiene la información del objeto a estudiar [59]. Sin embargo, las cámaras solo

capturan la información relacionada a la amplitud de la luz, lo que ocasiona la pérdida de la información de fase, esencial para reconstruir el campo óptico completo [60].

Para solucionar esta limitación, se emplea la técnica denominada Phase Shifting Holography (PSH), que consiste en tomar varias imágenes de interferencia mientras se introduce un desplazamiento controlado en la fase del haz de referencia. A partir de estas imágenes, es posible recuperar la información de fase perdida [10].

Sin embargo, registrar y almacenar varias imágenes con diferentes desplazamientos de fase puede ser costoso y complejo, especialmente cuando se trabaja con sistemas en tiempo real o de gran volumen de datos. Para abordar este desafío, se propone un enfoque que combina el uso del algoritmo Matching Pursuit (MP) y modelos generativos avanzados.

El algoritmo MP se utiliza como una técnica de preprocesamiento para eliminar ruido de las imágenes de interferencia, lo que mejora la calidad de las señales obtenidas y facilita su posterior análisis. El MP permite representar de manera dispersa las imágenes, lo que ayuda a identificar y preservar las características relevantes mientras se elimina información redundante o ruido.

Posteriormente, un modelo generativo se entrena para sintetizar imágenes desplazadas de fase a partir de una sola imagen de referencia. Este enfoque elimina la necesidad de registrar múltiples imágenes con desplazamientos físicos, ya que el modelo es capaz de generar hologramas con desplazamientos de fase virtuales. Además, estas imágenes sintéticas pueden ser utilizadas para recuperar la información de fase perdida, simplificando el proceso y reduciendo significativamente los costos asociados al almacenamiento y adquisición de datos.

Este trabajo propone la combinación de Matching Pursuit y modelos generativos como una alternativa eficiente y precisa para la recuperación de fase en holografía. Este enfoque no solo aprovecha el preprocesamiento para mejorar la calidad de las imágenes, sino que también elimina la necesidad de capturar físicamente múltiples hologramas desplazados, proporcionando una solución computacionalmente viable y escalable.

En estudios previos, se han explorado métodos avanzados como las redes neuronales convolucionales (CNNs) para problemas de recuperación de fase en holografía [61]. Sin embargo, no se han encontrado trabajos que combinen Matching Pursuit y modelos generativos para abordar este tipo de problemas. Por lo tanto, esta propuesta representa una contribución novedosa al campo,

con el potencial de optimizar tanto la adquisición de datos como el procesamiento de imágenes holográficas.

2.1. Eliminación de Ruido en Imágenes de Interferencia

La eliminación de ruido en imágenes de interferencia es un aspecto crucial en diversas aplicaciones científicas y tecnológicas, especialmente en la metrología óptica y la holografía cuántica. La presencia de ruido en estas imágenes puede degradar significativamente la calidad de la información obtenida, afectando la precisión en la recuperación de fase y en la interpretación de los datos. Para abordar este problema, se han desarrollado múltiples técnicas de filtrado y algoritmos de procesamiento de señales que buscan minimizar el ruido sin perder información relevante.

Las imágenes de interferencia pueden estar sujetas a diversas fuentes de ruido, incluyendo ruido gaussiano, ruido de disparo y ruido de cuantización. El ruido gaussiano surge de fluctuaciones aleatorias en la intensidad de la señal capturada por los sensores, mientras que el ruido de disparo está relacionado con la naturaleza discreta de la detección de fotones [12]. Por otra parte, el ruido de cuantización proviene de limitaciones en la resolución digital de los dispositivos de captura [62].

Existen diversas técnicas para la eliminación de ruido en imágenes de interferencia; entre las más utilizadas se encuentran los filtros espaciales, los métodos basados en transformadas y las técnicas de aprendizaje profundo:

- Filtros espaciales: Los filtros como el promedio y la mediana se han utilizado ampliamente para reducir el ruido en imágenes [63]. El filtro de mediana, en particular, es efectivo para eliminar ruido impulsivo sin afectar los bordes de la imagen.
- Transformada de Fourier y wavelets: La transformada de Fourier permite eliminar el ruido de alta frecuencia mediante el filtrado en el dominio de la frecuencia [64]. La transformada wavelet, por otro lado, facilita la eliminación de ruido adaptativa al descomponer la imagen en diferentes niveles de resolución.

- Aprendizaje profundo: Métodos basados en redes neuronales convolucionales han demostrado ser altamente efectivos en la eliminación de ruido en imágenes de interferencia. Estas redes pueden ser entrenadas con grandes conjuntos de datos para aprender patrones de ruido y eliminarlos de manera eficiente [65].

La eliminación de ruido en imágenes de interferencia tiene aplicaciones en la metrología óptica, la holografía y la imagenología biomédica. En el contexto de la holografía cuántica, la eliminación precisa del ruido es fundamental para la reconstrucción de fase en experimentos con luz no detectada [66]. Sin embargo, uno de los principales desafíos es equilibrar la reducción de ruido con la preservación de detalles importantes en la imagen.

2.2. Matching Pursuit como Método de Preprocesamiento

Matching Pursuit es un algoritmo de aproximación utilizado en el procesamiento de señales e imágenes, que permite descomponer una señal compleja en una combinación lineal de funciones o "átomos" de una base redundante, también conocida como diccionario [67]. La idea detrás de Matching Pursuit es encontrar de manera iterativa las mejores aproximaciones de la señal original usando una base de señales sencillas (generalmente senoidales, funciones de onda, o funciones con características específicas). Este proceso resulta útil en aplicaciones donde la señal puede ser compleja y es deseable representar solo sus componentes más significativos, facilitando la compresión o el análisis [68].

2.2.1. Fundamentos de Matching Pursuit

El algoritmo de Matching Pursuit fue propuesto inicialmente por Mallat y Zhang en 1993. Su objetivo principal es descomponer una señal x en una combinación de funciones o "átomos" de un diccionario D, que es un conjunto de funciones indexadas por un parámetro que describe las características del átomo (por ejemplo, su frecuencia, fase, o posición) [67]. Cada átomo del

diccionario tiene características particulares que ayudan a descomponer una señal compleja en componentes más simples y específicas [69].

En términos matemáticos, se busca una descomposición de la forma (Ecuación 2.1):

$$f = \sum_{n=0}^{N} \propto_n g_n + R_N \tag{2.1}$$

donde:

$\propto_n$ es el coeficiente de cada función (o átomo) del diccionario.

g_n es una función o vector en el diccionario D.

R_N es el residuo después de N iteraciones, que representa la parte de la señal que no ha sido explicada por los átomos seleccionados.

2.2.2. Funcionamiento del Algoritmo de Matching Pursuit

El algoritmo Matching Pursuit realiza una aproximación de la señal de manera iterativa, eligiendo en cada paso el átomo del diccionario que mejor se correlaciona con el residuo de la señal actual (la parte de la señal que aún no ha sido explicada) [70]. Los pasos que sigue el algoritmo son los siguientes:

1. Inicialización: Se empieza con la señal original f y un residuo inicial $R_0 = f$
2. Selección del Átomo Óptimo: En cada iteración, el algoritmo selecciona el átomo g_n que mejor correlaciona o "empareja" con el residuo actual R_n. Esto se realiza buscando el átomo que maximiza el producto escalar $\langle R_n, g \rangle$, lo cual equivale a proyectar el residuo sobre cada átomo del diccionario y escoger el que más se asemeje al residuo [67].
3. Actualización del Residuo: Una vez que se ha seleccionado el átomo g_n con el coeficiente, se actualiza el residuo restando la contribución del átomo seleccionado (Ecuación 2.2):

$$R_{n+1} = R_n - \propto_n g_n \tag{2.2}$$

Este paso asegura que el nuevo residuo contiene únicamente la parte de la señal que no fue explicada por los átomos ya seleccionados [68].

4. Iteración: Los pasos de selección de átomo y actualización del residuo se repiten iterativamente hasta alcanzar un criterio de parada, que puede ser un número máximo de iteraciones o cuando la energía (norma) del residuo cae por debajo de un umbral deseado [69].

2.2.3. Características y Propiedades

Matching Pursuit tiene varias propiedades que lo hacen valioso en el procesamiento de señales e imágenes:

- Matching Pursuit tiende a seleccionar pocos átomos significativos, lo que da lugar a una representación dispersa y eficiente de la señal [71].
- La descomposición se adapta automáticamente a las características de la señal, capturando patrones específicos como picos, transiciones y características de frecuencia [70].
- Se pueden usar diccionarios grandes y redundantes con diferentes tipos de átomos (senoides, wavelets, Gaussianas, entre otros) para capturar mejor las características de la señal [69].
- Aunque Matching Pursuit es computacionalmente intensivo, es menos costoso que métodos de búsqueda exhaustiva o de optimización global en diccionarios grandes [68].

2.2.4. Generación de Diccionarios para Matching Pursuit

El algoritmo de Matching Pursuit (MP) es una técnica de aproximación iterativa utilizada para representar una señal como una combinación dispersa de funciones base seleccionadas de un diccionario determinado [67]. La construcción de este diccionario es un aspecto fundamental en el rendimiento del algoritmo, ya que define la base sobre la cual se proyecta la señal y determina la calidad de la representación dispersa.

Un diccionario en Matching Pursuit es un conjunto de funciones denominadas "átomos" que pertenecen a un espacio de funciones predefinido. Estas funciones pueden ser ortonormales, como las bases de Fourier o de wavelets, o pueden ser redundantes, permitiendo representaciones más flexibles, pero computacionalmente más costosas [72].

El objetivo de este diccionario es proporcionar un repertorio suficientemente amplio de átomos para que cualquier señal pueda aproximarse con un número reducido de coeficientes. Para ello, se utilizan estrategias de diseño del diccionario que pueden clasificarse en diccionarios predefinidos y diccionarios aprendidos.

Los diccionarios predefinidos están compuestos por funciones matemáticamente establecidas, diseñadas para capturar características específicas de las señales. Algunos de los diccionarios más comunes sobresalen:

- El Diccionario de Fourier, adecuado para representar señales periódicas o cuasi-periódicas [73].
- El Diccionario de Wavelets, excelente para representar señales con estructuras jerárquicas o transitorias [72].
- Los Diccionarios Gabor, optimizados para análisis en el dominio tiempo-frecuencia [74].

Estos diccionarios están ampliamente documentados y permiten implementar Matching Pursuit de manera eficiente. Sin embargo, pueden no ser óptimos para señales específicas cuya estructura no se ajusta a sus bases.

Para mejorar la representación de señales complejas, se emplean algoritmos de aprendizaje de diccionarios, como Dictionary Learning o K-SVD [75]. Estos algoritmos optimizan la selección de los átomos basándose en un conjunto de entrenamiento, permitiendo una mejor adaptación a la naturaleza específica de las señales procesadas.

El proceso de aprendizaje de diccionarios generalmente se basa en los siguientes pasos:

- Inicialización: se parte de un diccionario aleatorio o basado en un conjunto de funciones conocidas.
- Actualización de Representaciones: para cada señal en el conjunto de entrenamiento, se aplica Matching Pursuit o OMP (Orthogonal Matching Pursuit) para obtener una representación dispersa.

- Optimización del Diccionario: se ajustan los átomos utilizando métodos como K-SVD, que refina iterativamente la estructura del diccionario.

Los diccionarios aprendidos han demostrado ser particularmente útiles en tareas de procesamiento de imágenes y señales biomédicas, donde las estructuras son altamente no lineales y variables [76].

La selección del diccionario adecuado depende del tipo de señal y de las restricciones computacionales. Los diccionarios predefinidos son eficientes y rápidos, pero pueden ser subóptimos para ciertos tipos de datos. En cambio, los diccionarios aprendidos proporcionan mayor flexibilidad, pero a costa de un mayor costo computacional [77].

2.2.5. Aplicaciones y Limitaciones en el Preprocesamiento de Imágenes

Matching Pursuit se ha utilizado en diversas aplicaciones, entre ellas:

- La representación dispersa permite comprimir señales o imágenes al guardar solo los átomos más relevantes [67,69].
- Al seleccionar solo los átomos con mayores correlaciones, se puede reducir el ruido en la señal al evitar átomos que representan el ruido [78].
- En el procesamiento de imágenes, permite descomponer la imagen en componentes específicos, facilitando la extracción de características para el reconocimiento de patrones [79].
- Para imágenes de interferencia con desplazamiento en fase, Matching Pursuit ayuda a analizar patrones específicos en cada imagen [80].

2.2.6. Desventajas y Limitaciones

A pesar de sus beneficios, Matching Pursuit también tiene algunas limitaciones:

- Alto costo computacional debido a que la búsqueda del átomo óptimo en cada iteración puede ser intensiva en cómputo, especialmente cuando se trata de diccionarios grandes y redundantes [68].
- La calidad de la representación depende en gran medida de la elección del diccionario. Si el diccionario no contiene átomos que se asemejen a las características de la señal, el rendimiento puede verse afectado [81].
- En ciertos casos, Matching Pursuit puede seleccionar demasiados átomos, ajustándose al ruido de la señal y no a sus componentes fundamentales [70].

2.2.7. Variantes y Optimización

Para superar algunas de las limitaciones listadas anteriormente, se han desarrollado variantes de Matching Pursuit como:

- Orthogonal Matching Pursuit (OMP), mejora que garantiza que todos los átomos seleccionados sean ortogonales al residuo en cada iteración, reduciendo el número de átomos necesarios [68].
- Stagewise OMP (StOMP), introduce criterios de parada en el proceso de selección de átomos para acelerar el proceso y evitar sobreajuste [82].
- Approximate MP (AMP), realiza aproximaciones para reducir el costo computacional, útil en diccionarios muy grandes [83].

2.2.8. Relevancia del Preprocesamiento Propuesto

El preprocesamiento de imágenes constituye una etapa crítica en la generación de hologramas sintéticos con desplazamiento de fase, particularmente cuando estas imágenes son empleadas como datos de entrada para modelos de aprendizaje profundo. Las imágenes de interferencia suelen estar afectadas por diversas fuentes de ruido (instrumental, ambiental y electrónico) que

degradan la calidad de las franjas de interferencia y, en consecuencia, afectan la extracción de información relevante asociada a la fase óptica.

En este contexto, el uso de Matching Pursuit (MP) como técnica de preprocesamiento permite abordar de manera directa varias de las limitaciones inherentes a los datos experimentales. Al representar las imágenes mediante una combinación dispersa de átomos seleccionados de un diccionario, MP posibilita la atenuación del ruido sin destruir las estructuras interferométricas relevantes, preservando patrones espaciales esenciales para la posterior reconstrucción holográfica.

La relevancia de esta propuesta radica en que el preprocesamiento mediante MP no se limita a una mejora visual de las imágenes, sino que actúa como un mecanismo de regularización implícita sobre los datos de entrada. Al reducir componentes no informativas y enfatizar características estructurales dominantes, se favorece una representación más estable y consistente, lo cual resulta importante cuando las imágenes son utilizadas para entrenar modelos generativos. De este modo, el modelo de aprendizaje profundo puede concentrarse en aprender relaciones físicas significativas, en lugar de ajustarse a variaciones inducidas por el ruido.

Asimismo, la generación y optimización de diccionarios específicos para este tipo de imágenes permite adaptar el proceso de preprocesamiento a las características propias de los hologramas con desplazamiento de fase. Esto representa una ventaja frente a métodos de filtrado tradicionales, los cuales suelen operar de manera global y pueden eliminar información relevante junto con el ruido. En contraste, el enfoque basado en MP ofrece flexibilidad en la selección de átomos, el control del nivel de aproximación y la posibilidad de optimizar el compromiso entre reducción de ruido y preservación de detalles.

En conjunto, el preprocesamiento propuesto establece una base sólida para la construcción del modelo de aprendizaje profundo presentado en capítulos posteriores. Su impacto se manifiesta en la calidad de los datos de entrenamiento y en la estabilidad del proceso de aprendizaje, aspectos que resultan determinantes para la generación de hologramas sintéticos con mayor fidelidad.

CAPÍTULO 3

GENERACIÓN DE IMÁGENES CON MODELOS DE APRENDIZAJE PROFUNDO

3.1. Redes Neuronales en Procesamiento de Imágenes

3.1.1. Red Neuronal Multicapa

En la era digital actual, la inteligencia artificial (IA) y el aprendizaje automático (Machine Learning) han transformado una amplia variedad de industrias. Entre los componentes fundamentales y ampliamente utilizados en estos campos se encuentran las redes neuronales artificiales. Dentro de este grupo, las Redes Neuronales de Perceptrón Multicapa, conocidas como MLP por sus siglas en inglés (Multi-Layer Perceptron), son particularmente relevantes debido a su capacidad para resolver problemas complejos de clasificación y regresión [84].

El concepto de redes neuronales artificiales (Figura 3.1) se inspira en la estructura y el funcionamiento del cerebro humano y su estructura se separa por capas de la siguiente forma:

- Capa de entrada o axón de entrada – vía por donde los datos acceden a la neurona.
- Sinapsis - encargadas de cargar los pesos.
- Dendritas - la conexión entre los valores de la sinapsis a la neurona.
- Cuerpo o neurona - donde recibe la suma ponderada y una función de activación.
- Cuello de axón - encargada de llevar el resultado final a una salida.
- Axón de salida - es la capa de salida que dará el resultado final.

Los pesos son valores numéricos que empiezan siendo aleatorios y existe uno por cada valor de entrada, es decir, si se tienen n valores de entrada y k pesos entonces $n = k$. Al pasar por la sinapsis, se realiza el producto de los valores de entrada con los pesos; al pasar a la neurona se aplica una función de activación σ a la suma ponderada de estos productos (Ecuación 3.1).

$$y = \sigma(\sum_{i=1}^{n} w_i x_i) \tag{3.1}$$

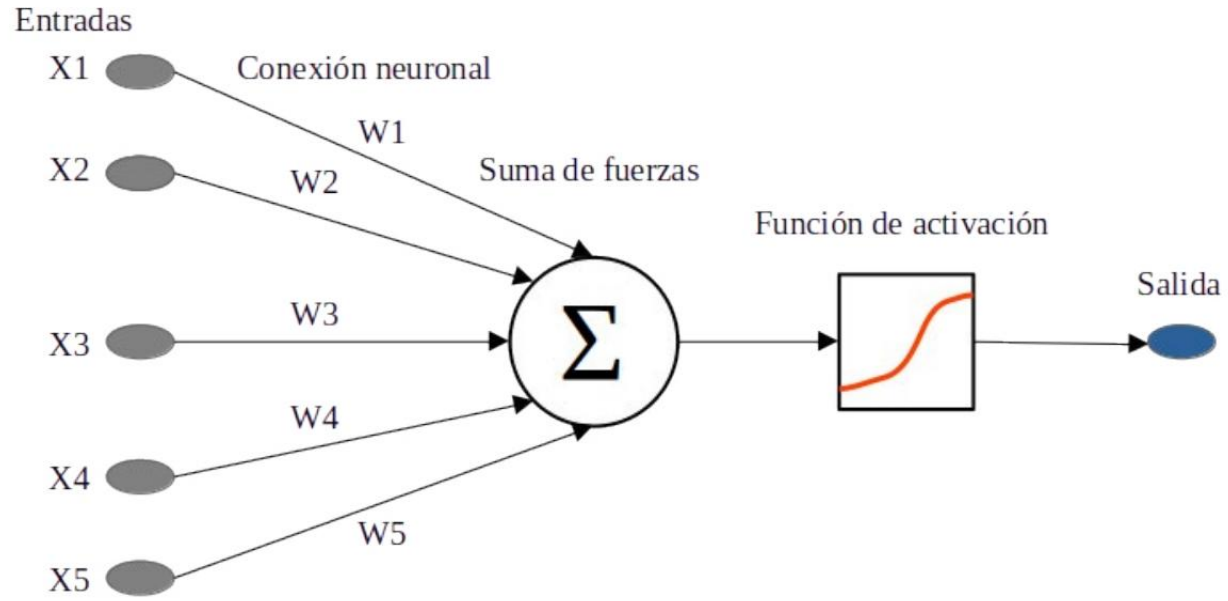


Figura 3.1 Arquitectura de una red neuronal artificial.

Una red MLP es un tipo de red neuronal artificial "feedforward", es decir, en la que las conexiones entre las neuronas no forman ciclos. Se conforman por una capa de entrada, una o más capas ocultas y una capa de salida. Cada neurona recibe varias entradas y, como se muestra en la Figura 3.1, combina estos valores mediante una suma ponderada determinada por los pesos asociados a cada conexión. El resultado de esta combinación se transforma posteriormente mediante una función de activación, y envían los resultados a las siguientes capas [85].

El perceptrón, desarrollado por Frank Rosenblatt en 1958, es el precursor de las redes MLP. Sin embargo, el perceptrón simple tiene limitaciones significativas, ya que solo puede resolver problemas linealmente separables. Para superar esta limitación, se introdujeron las capas ocultas en el MLP, lo que permitió a la red aprender representaciones más complejas y resolver problemas no lineales [86].

El funcionamiento de una red MLP se puede dividir en dos fases principales: la fase de entrenamiento y la fase de inferencia. Durante la fase de entrenamiento, la red ajusta sus pesos y

sesgos internos para minimizar el error en la predicción, utilizando un proceso conocido como retropropagación (backpropagation). Este algoritmo calcula el gradiente del error con respecto a los pesos, lo que permite actualizarlos en la dirección que minimiza el error, siguiendo el método del descenso del gradiente [87].

Las funciones de activación juegan un papel crucial en las redes MLP, permitiendo a la red aprender y modelar relaciones no lineales entre las entradas y las salidas. Algunas de las funciones de activación más comunes incluyen la función sigmoide, la tangente hiperbólica (tanh) y la rectificada lineal (ReLU). Cada una tiene características particulares que pueden influir en la convergencia y el rendimiento del modelo.

Las redes MLP tienen una amplia gama de aplicaciones debido a su flexibilidad y capacidad para modelar patrones complejos; en la visión por computadora, se emplean para tareas como la clasificación de imágenes y la detección de objetos, en procesamiento de lenguaje natural (NLP), se utilizan para el análisis de sentimientos, traducción automática y reconocimiento de voz. Además, las MLP son fundamentales en áreas como la predicción de series temporales, el análisis de datos financieros (evaluación de riesgos, predicción de tendencias de mercado y automatización del trading) y la biomedicina (predicción de enfermedades, diagnóstico de condiciones a partir de imágenes médicas y personalización de tratamientos) [84].

Las redes MLP no están exentas de limitaciones, una de las principales es que tienden a ser modelos de caja negra, lo que significa que es difícil interpretar cómo toman decisiones. Esto puede ser problemático en aplicaciones donde la explicabilidad es crucial, como en el ámbito médico o legal. Otra limitación es la necesidad de una gran cantidad de datos y poder de cómputo para entrenar modelos efectivos; si bien las MLP pueden aprender patrones complejos, requieren muchos datos etiquetados para hacerlo de manera efectiva. Además, los modelos grandes (aquellos con un número elevado de parámetros debido a la presencia de muchas capas, un gran número de neuronas por capa o ambas) pueden ser propensos al sobreajuste (overfitting), donde el modelo se ajusta demasiado a los datos de entrenamiento y falla al generalizar a nuevos datos [84].

3.1.2. Aprendizaje Profundo

El Deep Learning, subcategoría del aprendizaje automático (Machine Learning), ha revolucionado el campo de la inteligencia artificial (IA) y la ciencia de datos. Se basa en redes neuronales artificiales por lo que este enfoque permite a los sistemas aprender representaciones jerárquicas de datos mediante múltiples capas de abstracción. La capacidad del Deep Learning para resolver problemas complejos en visión por computadora, procesamiento del lenguaje natural, y otras áreas, ha generado un avance sin precedentes en la automatización y el análisis de grandes volúmenes de datos [88].

El término "Deep" hace referencia a la profundidad de las capas en una red neuronal; mientras que una red neuronal tradicional puede tener solo unas pocas capas, un modelo de Deep Learning puede tener decenas o cientos de ellas. Dicha profundidad se logra apilando múltiples capas de diferentes tipos (convoluciones, de pooling o densas), lo que permite modelar relaciones complejas no lineales y aprender características abstractas de los datos. Esto es crucial para tareas como la clasificación de imágenes, el reconocimiento de voz o la traducción automática [84].

En particular, el Deep Learning se sustenta en el uso de redes neuronales convolucionales (CNNs) para procesamiento de imágenes y redes neuronales recurrentes (RNNs) para secuencias temporales, como texto o audio. Las CNNs explotan la estructura local de los datos, por medio de filtros (o kernels) que se deslizan sobre la imagen de entrada para detectar patrones como bordes o texturas, generando un mapa de características (feature map). Un principio fundamental es la compartición de parámetros, donde un mismo filtro se aplica repetidamente en toda la imagen. Esta técnica reduce drásticamente el número de parámetros a aprender y confiere a la red la capacidad de invarianza a la traslación, es decir, que puede reconocer un patrón sin importar dónde se encuentre la imagen [2].

La razón por la que las CNNs son modelos de Deep Learning es que la profundidad de las capas convolucionales crea una jerarquía de características. Las primeras capas aprenden a reconocer elementos para detectar partes de objetos (como ojos o ruedas); y las capas más profundas usan estas combinaciones para identificar los objetos completos de alta complejidad (rostros, animales o vehículos). Esta capacidad de construir características complejas a partir de unas simples es lo que hace al Deep Learning una herramienta eficaz en el análisis visual. Por su parte, las RNNs aprovechan la dependencia temporal de los datos secuenciales, como palabras en una oración [2].

El Deep Learning ha demostrado ser altamente eficaz en una amplia variedad de aplicaciones, desde la detección de objetos en imágenes hasta la conducción autónoma. Algunas de sus principales ventajas incluyen la capacidad de manejar datos no estructurados, la reducción de la necesidad de ingeniería de características y su desempeño superior en problemas complejos.

No obstante, el Deep Learning también enfrenta varios desafíos. Las redes profundas requieren miles o millones de ejemplos etiquetados para entrenarse adecuadamente, lo que no siempre está disponible en todas las disciplinas. Además, el entrenamiento de redes neuronales profundas es computacionalmente costoso, requiriendo hardware especializado como GPUs o TPUs [84].

3.2. Autoencoders para la Generación de Imágenes

La generación de imágenes mediante inteligencia artificial ha experimentado un crecimiento notable en los últimos años, en parte gracias al desarrollo y la popularización de modelos de aprendizaje profundo. Entre estos, los autoencoders (AE) se destacan como una de las arquitecturas más efectivas para aprender representaciones compactas y útiles de datos. Se utilizan en diversas aplicaciones como la compresión de imágenes, la reducción de ruido y la generación de imágenes.

3.2.1. Fundamentos de los Autoencoders

Un autoencoder es una red neuronal diseñada para aprender una representación de los datos de entrada en un espacio de menor dimensión y luego reconstruir esos datos a partir de esa representación. Esta red consta de dos partes principales: el encoder (codificador) y el decoder (decodificador). El encoder toma la entrada, generalmente de alta dimensión (como una imagen), y la mapea a un espacio de menor dimensión, conocido como el espacio latente o código. El decoder, por su parte, toma esta representación comprimida y trata de reconstruir la entrada original [89].

La eficacia de los autoencoders se basa en la capacidad de capturar las características más relevantes de los datos mientras descarta la información menos significativa, lo que los hace especialmente útiles para tareas de compresión y generación.

3.2.2. Autoencoders para la Generación de Imágenes

Los autoencoders pueden utilizarse para la generación de imágenes en el contexto de modelos generativos. En estos casos, el objetivo no es solo reconstruir las imágenes de entrada, sino generar nuevas imágenes a partir de un espacio latente. Para lograrlo, es necesario entrenar la red para que el espacio latente tenga una estructura que permita muestrear nuevas imágenes de forma efectiva. Esto se logra, generalmente, mediante el uso de variantes de los autoencoders, como los variational autoencoders (VAE) [90].

El VAE es una extensión del autoencoder clásico que introduce una distribución probabilística en el espacio latente. En lugar de aprender un único punto en el espacio latente para cada entrada, el VAE modela una distribución de probabilidad (normalmente una distribución gaussiana) sobre el espacio latente. Durante el proceso de entrenamiento, el modelo intenta minimizar la diferencia entre la distribución de probabilidad generada por el encoder y la distribución estándar del espacio latente, utilizando una técnica conocida como variational inference [90].

La principal ventaja de los VAE en la generación de imágenes es que, al modelar el espacio latente de esta manera probabilística, el VAE puede generar nuevas imágenes muestreando puntos aleatorios de esta distribución. Este enfoque permite la creación de nuevas imágenes que no solo son similares a las imágenes de entrenamiento, sino también creativas y originales, lo que los convierte en una herramienta poderosa en la síntesis de imágenes [90].

En lugar de usar redes completamente conectadas, los autoencoders convolucionales (CAE) emplean redes neuronales convolucionales tanto en el encoder como en el decoder. Este tipo de autoencoder es particularmente eficaz para trabajar con imágenes, ya que las redes convolucionales son buenas para capturar patrones espaciales y texturales en las imágenes [91].

En el contexto de la generación de imágenes, los autoencoders convolucionales se utilizan para aprender una representación de alta calidad de las imágenes y luego generar nuevas imágenes a partir de estas representaciones. Al igual que los VAE, los CAE también pueden ser modificados para ser generativos, produciendo imágenes realistas a partir de puntos en el espacio latente [91].

Los VAE y los CAE se utilizan en la creación de imágenes realistas de alta calidad. Al entrenar estos modelos con grandes conjuntos de datos de imágenes, los modelos aprenden las características generales de las imágenes en un dominio específico, como rostros humanos, paisajes u objetos. A partir de esto, se pueden generar imágenes completamente nuevas que tengan una estructura coherente y visualmente atractiva [90,91].

Los autoencoders también se utilizan en la restauración de imágenes, como la eliminación de ruido o la superresolución. Al ser entrenados con imágenes de baja calidad o con ruido, los autoencoders aprenden a mapear estas imágenes a un espacio latente y luego a reconstruirlas de manera más nítida y clara. Esto ha sido aplicado con éxito en la mejora de imágenes médicas, imágenes satelitales y en la industria del cine y la fotografía para mejorar la calidad visual de imágenes deterioradas [91].

3.2.3. Arquitectura de un Autoencoder

La arquitectura de un autoencoder (AE) está compuesta por tres partes principales: el encoder (codificador), el espacio latente y el decoder (decodificador). Estas partes trabajan en conjunto para aprender una representación comprimida de los datos de entrada y luego reconstruirlos.

3.2.3.1. Encoder (Codificador)

El encoder es la primera parte del AE, cuyo objetivo es mapear los datos de entrada a una representación de menor dimensión en el espacio latente. Este proceso de mapeo se realiza a través de una serie de capas neuronales, que generalmente son capas densas o capas convolucionales, dependiendo de la naturaleza de los datos (imágenes, texto, etc.). La idea principal es que el

encoder reduzca la dimensionalidad de los datos, extrayendo las características más relevantes mientras descarta la información menos significativa [89].

3.2.3.1.1 Entrada

La entrada del AE puede ser una imagen, un vector de características o cualquier otro tipo de datos. La entrada x tiene una dimensión D que corresponde a las características originales del dato.

3.2.3.1.2. Capas del Encoder

El encoder está compuesto por varias capas de transformación, y pueden ser:

- Capas completamente conectadas donde cada neurona en una capa está conectada a todas las neuronas de la capa siguiente. Estas capas permiten transformar los datos a un espacio de menor dimensión [84].
- Capas convolucionales (en el caso de imágenes) en el que las redes convolucionales son útiles para identificar patrones locales y características espaciales dentro de las imágenes. La salida de estas capas son mapas de activación que ayudan a representar características como bordes, texturas y formas dentro de una imagen [88].
- Funciones de activación, luego de cada transformación lineal, se aplica una función de activación como ReLU, sigmoide o tangente hiperbólica. Estas funciones permiten introducir no linealidades que ayudan a modelar relaciones complejas en los datos [92].

3.2.3.2. Espacio Latente

El espacio latente es el punto intermedio entre el encoder y el decoder, y constituye la representación comprimida de los datos de entrada.

La última capa del encoder mapea la entrada a un espacio latente de menor dimensión z. El tamaño del espacio latente es generalmente mucho menor que el de la entrada, lo que obliga al modelo a aprender las características más esenciales y relevantes de los datos [89].

En un autoencoder tradicional, z es un vector determinista, mientras que en una variante como el variational autoencoder (VAE), la representación latente z sigue una distribución probabilística. En el caso de los VAE, el espacio latente no solo contiene una representación comprimida, sino que se modela como una distribución de probabilidad (normalmente una distribución gaussiana). Esto permite generar nuevos puntos latentes a partir de muestras aleatorias de esta distribución, lo que facilita la generación de nuevos datos [90].

3.2.3.3. Decoder (Decodificador)

El decoder es la parte del AE que toma la representación comprimida z y trata de reconstruir la entrada original. La reconstrucción es una versión aproximada de los datos originales, aunque no idéntica debido a la compresión aplicada en el encoder [84].

3.2.3.3.1. Capas del Decoder

Al igual que el encoder, el decoder puede estar compuesto por capas completamente conectadas o capas convolucionales, dependiendo de la naturaleza de los datos. La diferencia clave es que, en el decoder, las transformaciones aumentan gradualmente la dimensionalidad hasta llegar a la misma dimensión que la entrada original [89].

En el caso de imágenes, el decoder a menudo utiliza capas transpuestas convolucionales para reconstruir la imagen. Estas capas "expanden" los mapas de activación generados en las capas anteriores del decoder, recuperando la resolución original de la imagen [84].

El decoder utiliza una función de activación adecuada para reconstruir la salida. Si la entrada es una imagen en escala de grises, se puede utilizar una activación sigmoide o lineal en la capa final.

3.2.3.3.2. Reconstrucción

La salida final del decoder es una versión reconstruida de la entrada original. La calidad de la reconstrucción depende de cuán efectivamente el encoder ha aprendido a representar las características esenciales de la entrada y de cuán bien el decoder puede descomprimir esta representación para generar una salida fiel.

3.2.4. Entrenamiento del Autoencoder

Durante el proceso de entrenamiento, el objetivo es minimizar la función de pérdida, que mide la diferencia entre la entrada original x y la salida reconstruida $\hat{x}$. La función de pérdida comúnmente utilizada es el error cuadrático medio (MSE), que calcula la diferencia cuadrada promedio entre los valores de la entrada y la reconstrucción:

$$L(x, \hat{x}) = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{x}_i)^2 \tag{3.2}$$

donde N es el número total de características de la entrada.

En el caso de los variational autoencoders (VAE), la función de pérdida incluye un término adicional que mide la diferencia entre la distribución latente aprendida y la distribución normal estándar. Esto ayuda a regularizar el espacio latente, asegurando que sea bien estructurado para la generación de datos [90].

CAPÍTULO 4

MÉTRICAS PARA LA EVALUACIÓN DE RESULTADOS

4.1. Métricas de Calidad de Imagen

Las métricas de calidad de imagen son herramientas cruciales para medir la fidelidad de una imagen, especialmente cuando se aplica un proceso de compresión, transmisión o manipulación de la imagen.

La calidad de una imagen puede definirse como la medida en la que la imagen representada refleja fielmente el contenido original. En términos generales, una imagen de alta calidad conserva los detalles, los colores y las texturas tal como se perciben en el mundo real. Sin embargo, en la práctica, la calidad de una imagen puede degradarse debido a diversos factores como la compresión, el ruido, la distorsión o los defectos de la captura [93].

La evaluación de la calidad de una imagen es crucial en diversas aplicaciones. Por ejemplo, en la compresión de imágenes, es importante reducir el tamaño de un archivo sin perder demasiada información visual. En áreas como la medicina, una imagen de baja calidad puede llevar a diagnósticos incorrectos, mientras que en la fotografía o en los medios de comunicación, la calidad visual de una imagen puede influir en la percepción y la estética del producto final [94]. En la visión por computadora, las métricas de calidad de imagen son esenciales para evaluar el rendimiento de algoritmos de procesamiento de imágenes, como la mejora de imágenes o la recuperación de fase [95].

Las métricas objetivas son aquellas que proporcionan una evaluación cuantitativa de la calidad de la imagen, sin necesidad de intervención humana. Estas métricas se basan en la comparación de la imagen original con la imagen procesada, y a menudo se utilizan para evaluar la eficacia de los algoritmos de compresión, restauración o mejora de imágenes.

4.1.1. MSE (Mean Squared Error)

El Error Cuadrático Medio (Mean Squared Error, MSE) es una métrica utilizada para cuantificar la diferencia entre una imagen de referencia X y una imagen procesada Y. Se calcula como la media de los cuadrados de las diferencias entre los valores de los píxeles de ambas imágenes y matemáticamente definida por la Ecuación 4.1:

$$MSE = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}[X(i,j) - Y(i,j)]^2 \tag{4.1}$$

donde M y N son el número de filas y columnas de la imagen respectivamente, mientras que $X(i,j)$ representan la intensidad del píxel en la posición (i,j) de la imagen original y $Y(i,j)$ la intensidad la imagen alterada[96].

El MSE tiene la ventaja de ser fácil de calcular, pero no siempre representa de manera precisa la percepción visual de la degradación en una imagen, ya que no considera factores como la estructura o el contraste local [97].

4.1.2. PSNR (Peak Signal-to-Noise Ratio)

El Pico de Relación Señal-Ruido (Peak Signal-to-Noise Ratio, PSNR) es una métrica derivada del MSE que expresa la calidad de una imagen en términos de decibeles (dB).

Se calcula a partir del MSE y mide la relación entre la potencia máxima de la señal y la potencia del ruido (Ecuación 4.2). Un PSNR más alto indica una menor distorsión y, por lo tanto, una mayor calidad de la imagen. En el contexto de una imagen digital típica (8 bits), los valores entre 30 y 40 dB son generalmente considerados de buena a excelente calidad, mientras que, en aplicaciones de alta precisión como la holografía o el análisis de patrones, a menudo se buscan valores superiores a 45 o 50 dB [98].

$$PSNR = 10\log_{10}\left(\frac{L^2}{MSE}\right) \tag{4.2}$$

El PSNR se utiliza ampliamente en la evaluación de calidad de imágenes comprimidas y restauradas, ya que proporciona una medida relativa de la distorsión. Un valor de PSNR más alto indica que la imagen procesada es más similar a la original. Sin embargo, al igual que el MSE, el PSNR no siempre correlaciona bien con la percepción visual humana, ya que no considera factores estructurales de la imagen [97].

4.1.3. SSIM (Structural Similarity Index)

A diferencia del MSE y PSNR, el Índice de Similitud Estructural (Structural Similarity Index, SSIM) intenta modelar la percepción humana de la calidad visual. Este índice considera la luminancia, el contraste y la estructura de la imagen. SSIM compara localmente la estructura de la imagen original y la imagen procesada, proporcionando una evaluación más precisa de la calidad percibida por el ojo humano [97].

Se basa en la idea de que el sistema visual humano es más sensible a cambios en la estructura de una imagen que a diferencias en la intensidad de los píxeles individuales.

El SSIM se define matemáticamente como:

$$SSIM(X,Y) = [l(X,Y)]^{\alpha} \cdot [c(X,Y)]^{\beta} \cdot [s(X,Y)]^{\gamma} \tag{4.3}$$

Donde:

- $l(X,Y)$ mide la similitud en la luminancia,
- $c(X,Y)$ mide la similitud en el contraste,
- $s(X,Y)$ mide la similitud en la estructura, y
- α, β, γ son parámetros que ponderan la importancia de cada componente (Wang et al., 2004).

El SSIM varía entre -1 y 1, donde un valor de 1 indica imágenes idénticas. A diferencia del PSNR y el MSE, el SSIM ha demostrado una correlación mucho mayor con la percepción visual humana,

por lo que es ampliamente utilizado en aplicaciones donde la calidad visual es fundamental, como la transmisión y compresión de imágenes [99].

CAPÍTULO 5

DESCRIPCIÓN E IMPLEMENTACIÓN DE LOS ALGORITMOS

En este capítulo se presenta la descripción de los algoritmos y procedimientos implementados a lo largo de este trabajo. Con el fin de proporcionar una visión general del flujo metodológico, la Figura 5.1 muestra un diagrama que resume las etapas principales del proceso, dividido en dos fases fundamentales: el preprocesamiento de los hologramas y el modelo de aprendizaje.

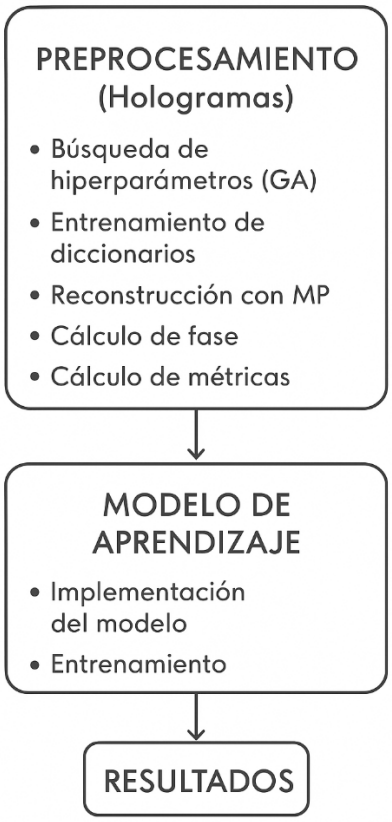


Figura 4.1 Representación esquemática del pipeline metodológico utilizado.

La primera fase, denominada "Preprocesamiento (Hologramas)", engloba todas las etapas necesarias para preparar las imágenes antes del entrenamiento del modelo. Esto incluye la búsqueda de hiperparámetros mediante algoritmos genéticos, el entrenamiento de diccionarios, la

reconstrucción de las imágenes con Matching Pursuit, y el cálculo tanto de la fase como de las métricas iniciales utilizadas para evaluar la calidad de las reconstrucciones.

Posteriormente, la fase de "Modelo de Aprendizaje" comprende la implementación y el entrenamiento del autoencoder propuesto para generar hologramas sintéticos a partir de las imágenes procesadas previamente. Esta etapa se centra exclusivamente en el diseño y ajuste del modelo.

Finalmente, los resultados obtenidos de ambas fases se integran para evaluar el desempeño general de la metodología propuesta. En las siguientes secciones se detallan los algoritmos utilizados, así como sus respectivas implementaciones.

5.1. Código del Algoritmo de Marching Pursuit

En esta sección se describen las distintas versiones del código desarrolladas para la implementación del método de Matching Pursuit (MP) optimizado mediante algoritmos genéticos (GA).

El objetivo de este bloque de trabajo fue encontrar una configuración óptima de los hiperparámetros asociados al proceso de reconstrucción de imágenes, garantizando una representación dispersa eficiente y una reducción significativa del error de reconstrucción.

Durante el desarrollo del proyecto se diseñaron y evaluaron tres versiones principales del código.La primera versión constituyó una implementación enfocada en validar la correcta interacción entre el Dictionary Learning y el MP.La segunda versión…

Finalmente, la tercera versión introdujo estrategias de paralelización para acelerar la evaluación de los individuos dentro del GA, aprovechando múltiples núcleos de procesamiento permitiendo un análisis comparativo más robusto.

Cada versión fue evaluada utilizando un conjunto de imágenes de referencia, considerando métricas cuantitativas como el error cuadrático medio (MSE) y cualitativas como la fidelidad visual de la reconstrucción.

El propósito de esta sección es exponer la evolución de las implementaciones, justificar las modificaciones realizadas y demostrar cómo dichas mejoras contribuyeron al desempeño global del sistema de optimización aplicado al Matching Pursuit.

5.1.1 Versión 1

Algoritmo 1. Optimización de parámetros de Matching Pursuit mediante Algoritmo Genético (Versión 1)

Entrada: Imagen de entrenamiento I, parámetros del algoritmo genético: tamaño de población N, número de generaciones G, probabilidad de cruce pc, probabilidad de mutación pm, rangos permitidos.

Salida: Conjunto óptimo de parámetros $\langle n\_components, n\_iter, \alpha \rangle$

1. Preprocesar I: convertir a escala de grises, redimensionar, normalizar y dividir en parches 32×32 con 50% de solapamiento.
2. Inicializar población de N individuos con parámetros $\langle n\_components, n\_iter, \alpha \rangle$.
3. Para cada generación $(1 \dots G)$:
    a) Evaluar cada individuo aplicando Matching Pursuit en todos los parches.
    b) Reconstruir la imagen y calcular el error (MSE).
    c) Seleccionar individuos, aplicar cruce con probabilidad pc y mutación con probabilidad pm.
    d) Restringir parámetros a sus rangos y aplicar elitismo
4. Devolver el individuo con menor error.

Esta primera versión del código (Algoritmo 1) tiene como propósito optimizar automáticamente los parámetros del algoritmo Matching Pursuit (MP) utilizando Dictionary Learning para la reconstrucción de imágenes. La optimización se realiza mediante un Algoritmo Genético (GA) implementando paralelización mediante multiprocessing.Pool para acelerar la evaluación de los individuos. Esta versión permite procesar imágenes de manera eficiente, explorando el espacio de parámetros [n_components, n_iter, alpha] para minimizar el error de reconstrucción.

El flujo de procesamiento inicia con la carga de imágenes en escala de grises, seguido de la normalización de los valores de píxel al rango [0,1]. Las imágenes pueden ser redimensionadas según el tamaño de análisis, lo que facilita la evaluación comparativa de distintas resoluciones.

Cada imagen se divide en parches de 32×32 píxeles con un 50% de solapamiento para permitir una reconstrucción localizada y controlada.

Cada parche se transforma al dominio de frecuencia mediante la Transformada Rápida de Fourier (FFT) y se centra mediante desplazamiento de frecuencia (fftshift). Posteriormente, se aplica Dictionary Learning para generar un diccionario disperso que representa cada parche, y se calcula el código disperso mediante el algoritmo de optimización seleccionado (omp o lars). La reconstrucción se obtiene multiplicando el código disperso por los componentes del diccionario y aplicando la transformada inversa, generando así cada parche reconstruido. Los parches se combinan promediando las zonas solapadas, generando la imagen completa, y se calcula el error cuadrático medio (MSE) respecto a la imagen original.

El Algoritmo Genético define individuos representando los parámetros [n_components, n_iter, alpha]. Se aplican operadores de selección por torneo, cruzamiento tipo Blend y mutación gaussiana, preservando el mejor individuo mediante elitismo. La evaluación de los individuos se realiza de manera paralela mediante multiprocessing.Pool, acelerando significativamente el cálculo del fitness para grandes poblaciones. Finalmente, los mejores parámetros son reportados y pueden ser guardados para análisis comparativo y reproducibilidad.

Tabla 5.1 Resumen de las funciones y bloques principales de la primera versión del código, incluyendo su propósito y las líneas de código correspondientes, para describir la arquitectura y flujo de procesamiento.

Bloque / Función	Descripción	Responsabilidad dentro del flujo	Líneas de código representativas
Carga y normalización	Convierte imágenes a escala de grises y normaliza valores	Preparar imágenes para análisis	img = Image.open(img_path).convert("L") img_array = normalize_image(np.array(img))
Redimensionamiento	Ajusta imágenes a tamaño predefinido	Analizar impacto de la resolución	img = resize_image(img)
División en parches solapados	Fragmenta imagen en parches 32×32 con 50% solapamiento	Facilita reconstrucción localizada	patches = process_in_patches(img_array, patch_size=(32, 32))
Transformación al dominio de frecuencia	FFT y centrado de frecuencias	Permite aplicar Dictionary Learning	freq_image = np.fft.fft2(image_array) freq_image_shifted = np.fft.fftshift(freq_image)
Dictionary Learning y MP	Entrena diccionario disperso y obtiene representación dispersa	Reconstrucción de parches optimizada	dict_learning = DictionaryLearning(...) sparse_code = dict_learning.fit_transform(flat_freq_image)
Reconstrucción de imagen completa	Combina parches promediando solapamientos	Genera imagen final reconstruida	reconstructed_image[i:i+32, j:j+32] += result_patch reconstructed_image /= count
Evaluación del error	Calcula MSE entre original y reconstruida	Cuantifica calidad de parámetros	error = np.mean((img_array - reconstructed_image)**2)
Algoritmo Genético (GA)	Define individuos y aplica operadores genéticos	Optimización automática de parámetros	toolbox.register("mate", tools.cxBlend, alpha=0.5) toolbox.register("mutate", tools.mutGaussian, mu=0, sigma=0.1, indpb=0.2) toolbox.register("select", tools.selTournament, tournsize=3)
Paralelización	Evalúa individuos simultáneamente	Reduce tiempo de cómputo	pool = multiprocessing.Pool() toolbox.register("map", pool.map)
Ejecución y selección final	Itera generaciones y selecciona mejor individuo	Optimización final y reporte de resultados	best_individual = tools.selBest(population, 1)[0]

La Tabla 5.1 resume los bloques funcionales del código y su papel dentro del flujo de procesamiento. Se identifica desde la carga y normalización de imágenes, que prepara los datos para un análisis consistente, hasta la división en parches y la transformación al dominio de frecuencia para aplicar Dictionary Learning y Matching Pursuit, que constituyen el núcleo de la reconstrucción. La combinación de parches reconstruidos y el cálculo del error de reconstrucción permiten cuantificar la eficacia de los parámetros evaluados, mientras que la implementación de GA con operadores de selección, cruzamiento y mutación asegura la exploración eficiente del espacio de parámetros. La inclusión de paralelización mediante multiprocessing.Pool acelera la evaluación de individuos, haciendo que el algoritmo sea más escalable para grandes poblaciones y volúmenes de imágenes. En la columna de líneas representativas se incluyen fragmentos del código que ilustran la implementación de cada módulo, ofreciendo una referencia directa para entender la arquitectura y funcionamiento del sistema.

Esta versión del código prioriza eficiencia y paralelización, utilizando multiprocessing.Pool para acelerar la evaluación de los individuos del GA. La arquitectura modular permite separar claramente el preprocesamiento, la reconstrucción, la evaluación y la optimización, facilitando su

mantenimiento y actualización. La reconstrucción de imágenes se basa en parches solapados y FFT, con Dictionary Learning y MP como núcleo de la optimización. La optimización automatizada mediante GA explora el espacio de parámetros [n_components, n_iter, alpha] de forma eficiente, preservando siempre el mejor individuo mediante elitismo. Esta versión es flexible con respecto al tamaño de imagen, lo que permite analizar el impacto de distintas resoluciones sobre la reconstrucción y el tiempo de cómputo.

El código implementa un flujo de procesamiento modular que integra preprocesamiento de imágenes, división en parches, reconstrucción mediante MP y Dictionary Learning, evaluación de error y optimización automática mediante GA. La paralelización reduce significativamente el tiempo de evaluación, mejorando la eficiencia general del algoritmo. Su diseño garantiza compatibilidad, robustez y reproducibilidad, permitiendo su ejecución estable en entornos de supercómputo y asegurando resultados confiables para distintos tamaños de imagen y configuraciones de parámetros.

5.1.2 Versión 2

Algoritmo 2. Optimización de parámetros de Matching Pursuit mediante Algoritmo Genético (Versión 2 – LNS, Python 3.6.8, sin paralelización)

Entrada: imágenes de entrenamiento I, tamaño deseado de las imágenes S, número de generaciones G, tamaño de población P, número de particiones para validación cruzada K.

Salida: Conjunto óptimo de parámetros $\langle n\_components, n\_iter, \alpha, tol \rangle$

1. Preprocesar las imágenes: redimensionar a S y normalizar intensidades.
2. Definir la función de aptitud basada en Matching Pursuit y validación cruzada K-fold.
3. Definir el espacio de búsqueda de los parámetros $\langle n\_components, n\_iter, \alpha, tol \rangle$
4. Inicializar una población de P individuos con parámetros aleatorios.
5. Para cada generación $g = 1, \dots, G$:
   5.1 Evaluar la aptitud de cada individuo mediante MSE promedio.
   5.2 Seleccionar individuos por torneo.
   5.3 Aplicar cruzamiento y mutación.
   5.4 Actualizar la población usando elitismo.
6. Seleccionar el individuo con menor error.
7. Devolver y almacenar los parámetros óptimos.

Tomando como base la versión descrita anteriormente, esta versión del código (Algoritmo 2) fue adaptada a Python 3.6.8 para garantizar compatibilidad con las bibliotecas disponibles en el Laboratorio Nacional de Supercómputo del Sureste de México.

El sistema se organiza en módulos interconectados que permiten un flujo de procesamiento de extremo a extremo. Inicialmente, las imágenes se cargan y se convierten a escala de grises, normalizando los valores de píxel al rango de 0 a 1. A continuación, se redimensionan según el tamaño de imagen deseado para permitir el análisis comparativo entre distintas resoluciones. Cada imagen se divide en parches de 32×32 píxeles con un 50% de solapamiento, y a cada parche se le aplica una ventana de Hann bidimensional para suavizar los bordes, minimizando artefactos en la posterior reconstrucción.

Cada parche suavizado se transforma al dominio de frecuencia mediante la Transformada Rápida de Fourier (FFT) y se centra utilizando el desplazamiento de frecuencia. Sobre este dominio se aplica Dictionary Learning para entrenar un diccionario disperso que permite representar cada parche de manera comprimida. La reconstrucción se realiza multiplicando el código disperso por los componentes del diccionario y aplicando la transformada inversa, obteniendo así un parche reconstruido. Los parches reconstruidos se combinan promediando las zonas solapadas para generar la imagen completa, y se calcula el error cuadrático medio (MSE) respecto a la imagen original. Este procedimiento se complementa con validación cruzada, dividiendo el conjunto de imágenes en subconjuntos y evaluando el error promedio, asegurando la robustez de los parámetros optimizados.

El Algoritmo Genético define individuos que representan conjuntos de parámetros [n_components, n_iter, alpha, tol]. Se aplican operadores de selección por torneo, cruzamiento tipo Blend y mutación gaussiana, preservando el mejor individuo mediante elitismo. Los individuos se evalúan durante varias generaciones, ajustando iterativamente los parámetros para minimizar el error de reconstrucción. Finalmente, el sistema permite procesar imágenes de distintos tamaños y almacena los mejores parámetros en archivos .pkl, facilitando la reproducibilidad y el análisis comparativo.

La Tabla 5.2 presenta una descripción detallada de los principales bloques funcionales del código, mostrando cómo cada módulo contribuye al flujo general de procesamiento. En ella se identifica

desde la carga y normalización de imágenes, que asegura que todas las entradas estén en un formato homogéneo y apto para el análisis, hasta la división de las imágenes en parches solapados y la aplicación de la ventana de Hann, que preparan los datos para una reconstrucción más precisa. También se incluyen la transformación al dominio de frecuencia y la aplicación de Dictionary Learning junto con Matching Pursuit, que constituyen el núcleo del algoritmo de reconstrucción. La combinación de parches reconstruidos y el cálculo del error de reconstrucción permiten evaluar la calidad de los parámetros del GA, mientras que la validación cruzada garantiza robustez frente a distintos subconjuntos de imágenes. Finalmente, se detallan los bloques del Algoritmo Genético y la ejecución por tamaños con almacenamiento de resultados, indicando cómo se seleccionan, cruzan y mutan los individuos, y cómo se preservan los mejores resultados. En la columna adicional se muestran las líneas de código representativas, proporcionando referencias directas de la implementación.

Tabla 5.2 Resumen de las funciones y bloques principales de la segunda versión del código, con énfasis en la compatibilidad con Python 3.6.8 y las modificaciones introducidas respecto a la versión base.

Bloque / Función	Descripción	Responsabilidad dentro del flujo	Líneas de código representativas
Carga y normalización de imágenes	Convierte las imágenes a escala de grises y normaliza los valores al rango [0,1].	Preparar imágenes para análisis y asegurar consistencia.	img = Image.open(img_path).convert("L")
			img_array = normalize_image(np.array(img))
Redimensionamiento de imágenes	Ajusta las imágenes a un tamaño predefinido.	Permite evaluar el efecto del tamaño de imagen en la reconstrucción.	img = resize_image(img, max_size)
División en parches solapados	Fragmenta la imagen en parches de 32×32 píxeles con 50% de solapamiento.	Facilita reconstrucción localizada y eficiencia en Dictionary Learning.	patches = process_in_patches(img_array, patch_size=(32, 32))
Aplicación de ventana de Hann	Suaviza los parches para reducir artefactos de borde.	Mejora calidad de reconstrucción y evita discontinuidades.	patch_hann = apply_hann_window(patch)
Transformación al dominio de frecuencia	FFT y centrado de frecuencias de cada parche.	Permite aplicar Dictionary Learning en el dominio de frecuencia.	freq_image = np.fft.fft2(image_array)
			freq_image_shifted = np.fft.fftshift(freq_image)
Dictionary Learning y Matching Pursuit	Entrena diccionario disperso y obtiene código disperso para reconstrucción de parches.	Reconstruye parches individuales optimizando componentes dispersos.	dict_learning = DictionaryLearning(...)
			sparse_code = dict_learning.fit_transform(flat_freq_image)
Reconstrucción de la imagen completa	Combina los parches reconstruidos promediando los solapamientos.	Genera la imagen final reconstruida y asegura consistencia en regiones solapadas.	reconstructed_image[i:i+32, j:j+32] += result_patch
			reconstructed_image /= count
Evaluación del error de reconstrucción	Calcula MSE entre imagen original y reconstruida.	Cuantifica calidad de cada conjunto de parámetros del GA.	error = np.mean((img_array - reconstructed_image)**2)
Validación cruzada	Divide el conjunto de imágenes en subconjuntos y calcula error promedio.	Asegura robustez de los parámetros optimizados frente a distintas imágenes.	val_error = evaluate_images(val_sets, individual, max_size)
Algoritmo Genético (GA)	Define individuos [n_components, n_iter, alpha, tol], aplica selección, cruzamiento, mutación y elitismo.	Optimiza automáticamente los parámetros de MP para minimizar error.	toolbox.register("mate", tools.cxBlend, alpha=0.5)
			toolbox.register("mutate", tools.mutGaussian, mu=0, sigma=0.1, indpb=0.2)
			toolbox.register("select", tools.selTournament, tournsize=3)
Ejecución por tamaños y almacenamiento	Itera sobre distintos tamaños de imagen y guarda resultados en archivos .pkl.	Permite análisis comparativo y reproducibilidad de experimentos.	best_params = optimize_images(image_paths, max_size=size_value)
			pickle.dump(best_parameters_all, f)

La implementación está diseñada para ser compatible con Python 3.6.8, garantizando ejecución estable en entornos de supercómputo sin conflictos de dependencias. La modularidad de la arquitectura permite separar claramente el preprocesamiento, la reconstrucción, la evaluación y la optimización, facilitando el mantenimiento y la actualización de módulos individuales. La calidad de la reconstrucción se garantiza mediante la aplicación de la ventana de Hann y la validación cruzada, lo que asegura que los parámetros optimizados funcionen de manera consistente para diferentes subconjuntos de imágenes.

La optimización automatizada mediante GA permite explorar el espacio de parámetros de manera eficiente, preservando siempre el mejor individuo mediante elitismo. Además, el sistema es flexible en cuanto a la resolución de las imágenes, lo que permite analizar cómo el tamaño de las imágenes afecta la reconstrucción y el tiempo de cómputo. Entre las limitaciones se encuentra la evaluación secuencial de individuos, lo que hace que el proceso sea más lento comparado con versiones paralelizadas, y la dependencia de la memoria y el tiempo de cómputo al procesar grandes volúmenes de datos.

El código implementa una arquitectura robusta y modular que integra preprocesamiento de imágenes con ventana de Hann, reconstrucción de parches mediante MP y Dictionary Learning, evaluación mediante MSE con validación cruzada y optimización automática de parámetros con GA. Su diseño prioriza compatibilidad, robustez y reproducibilidad, permitiendo su ejecución estable en entornos de supercómputo con Python 3.6.8 y asegurando resultados confiables para distintas resoluciones de imagen.

5.1.3 Versión 3

Algoritmo 3. Implementación de Matching Pursuit con Optimización por Algoritmo Genético y Paralelización por Índices (Versión 3 – LNS, multiprocessing)

Entrada: Conjunto de rutas de imágenes agrupadas por índice (run0/imgk, ...,run19/imgk), tamaño deseado de las imágenes S, número de generaciones G, tamaño de población P, número de particiones para validación cruzada K, $lista\ de\ índices\ a\ procesar$ $\{k_{1,}k_{2,\dots}\}$.

Salida: Conjunto óptimo de parámetros $\langle n\_components, n\_iter, \alpha, tol\rangle$ para cada índice de imagen y para cada tamaño de reescalado.

1. Definir funciones auxiliares para:
   - redimensionar y normalizar imágenes,
   - dividir imágenes en parches solapados,
   - aplicar ventana de Hann,
   - aplicar Matching Pursuit en dominio frecuencial.
2. Definir la función de aptitud como el MSE promedio de reconstrucción obtenido mediante validación cruzada K-fold.
3. Definir el espacio de búsqueda de parámetros $\langle n\_components, n\_iter, \alpha, tol\rangle$

4. Inicializar una población de P individuos con parámetros aleatorios dentro de los rangos definidos.
5. Para cada generación $g = 1, \ldots, G$:
    1. Evaluar la aptitud de cada individuo mediante validación cruzada:
        - reconstruir cada imagen usando Matching Pursuit por parches,
        - calcular el MSE entre la imagen original y la reconstruida.
    2. Seleccionar individuos mediante torneo.
    3. Aplicar cruzamiento y mutación a los descendientes.
    4. Re-evaluar únicamente los individuos modificados.
    5. Construir la nueva población conservando los mejores individuos (elitismo).
6. Seleccionar el mejor individuo de la población final.
7. Repetir el proceso para cada:
    - tamaño de imagen considerado,
    - índice de imagen, procesando estos casos en paralelo.
8. Almacenar los parámetros óptimos obtenidos en un archivo .pkl

Esta versión (Algoritmo 3), a diferencia de versiones anteriores, incorpora la aplicación de una ventana de Hann sobre los parches de la imagen y paraleliza el procesamiento de subíndices de imágenes dentro de cada tamaño, optimizando el tiempo de cómputo.

El flujo de procesamiento comienza con la carga de imágenes en escala de grises y la normalización de los valores de píxel entre 0 y 1, asegurando uniformidad en los datos. Se permite redimensionar las imágenes a distintas resoluciones, facilitando el análisis comparativo entre distintos tamaños.

Cada imagen se divide en parches de 32×32 píxeles con un 50% de solapamiento, lo que permite una reconstrucción localizada y un tratamiento uniforme de la información de alta frecuencia. Antes de aplicar el algoritmo de Matching Pursuit, cada parche se multiplica por una ventana de Hann bidimensional para minimizar efectos de borde y suavizar transiciones en los parches solapados.

Los parches se transforman al dominio de frecuencia mediante la FFT y se centra la frecuencia con fftshift. Posteriormente, se realiza Dictionary Learning para obtener un diccionario disperso que represente cada parche y se calcula su código disperso mediante el algoritmo lasso_lars. La reconstrucción de cada parche se obtiene multiplicando el código disperso por los componentes

del diccionario y aplicando la transformada inversa, generando finalmente la reconstrucción de la imagen completa mediante promedio de los parches solapados.

La evaluación del desempeño de los parámetros se realiza mediante error cuadrático medio (MSE) entre la imagen original y la reconstruida. Para mejorar la robustez de la evaluación, se implementa validación cruzada por pliegues sobre los conjuntos de imágenes.

El Algoritmo Genético genera individuos que representan los parámetros [n_components, n_iter, alpha, tol]. Se utilizan operadores de selección por torneo, cruzamiento definido por la función custom_mate y mutación gaussiana. Se preserva siempre el mejor individuo mediante elitismo. La paralelización se aplica sobre los subíndices de las imágenes, de manera que cada subíndice se procesa de forma independiente en un pool de procesos, acelerando considerablemente la evaluación de grandes volúmenes de datos.

La Tabla 5.3 presenta los bloques funcionales del código y su papel dentro del flujo de procesamiento. Comienza con la carga, normalización y redimensionamiento de imágenes para garantizar datos consistentes. Luego, cada imagen se divide en parches solapados y se aplica una ventana de Hann para suavizar los bordes. La transformación al dominio de frecuencia y la posterior aplicación de Dictionary Learning y Matching Pursuit constituyen el núcleo de la reconstrucción. Cada parche reconstruido se combina promediando los solapamientos para generar la imagen completa. El cálculo del MSE proporciona la métrica de evaluación de cada individuo, mientras que la validación cruzada garantiza la robustez de la evaluación. Finalmente, el Algoritmo Genético explora el espacio de parámetros mediante selección, cruzamiento y mutación, y la paralelización sobre subíndices acelera significativamente la optimización para múltiples imágenes y tamaños.

Tabla 5.3 Resumen de las funciones y bloques principales de la tercera versión del código, destacando mejoras en paralelización, aplicación de ventana de Hann, validación cruzada y soporte para distintos tamaños de imagen.

Bloque / Función	***Descripción***	***Líneas de código representativas***
Carga y normalización	Convierte imágenes a escala de grises y normaliza los valores de píxel	img = Image.open(img_path).convert("L")
		img_array = normalize_image(np.array(img))
Redimensionamiento	Ajusta la imagen a un tamaño específico	img = resize_image(img, max_size)
División en parches solapados	Fragmenta imagen en parches 32×32 con 50% de solapamiento	patches = process_in_patches(img_array, patch_size=(32, 32))
Aplicación de ventana de Hann	Suaviza bordes y transiciones de los parches	patch_hann = apply_hann_window(patch)
Transformación al dominio de frecuencia	FFT y centrado de frecuencia	freq_image = np.fft.fft2(image_array)
		freq_image_shifted = np.fft.fftshift(freq_image)
Dictionary Learning y Matching Pursuit	Entrena diccionario disperso y calcula representación dispersa	dict_learning = DictionaryLearning(...)
		sparse_code = dict_learning.fit_transform(flat_freq_image.T)
Reconstrucción de imagen	Combina los parches reconstruidos y promedia solapamientos	reconstructed_image[i:i+32, j:j+32] += result_patch
		reconstructed_image /= np.maximum(count, 1e-8)
Evaluación del error	Calcula MSE entre imagen original y reconstruida	error = np.mean((img_array - reconstructed_image)**2)
Validación cruzada	Promedia errores por pliegues	val_error = evaluate_images(val_sets, individual, max_size)
Algoritmo Genético	Define individuos, selecciona, cruza y muta	toolbox.register("mate", custom_mate)
		toolbox.register("mutate", tools.mutGaussian, ...)
		toolbox.register("select", tools.selTournament, ...)
Paralelización	Evalúa subíndices de imágenes en paralelo	with multiprocessing.Pool(processes=len(image_indices)) as index_pool:
		results = index_pool.map(process_index, index_args)
Ejecución por tamaño	Procesa cada tamaño secuencialmente y subíndices en paralelo	result_size_name, params_dict = run_for_size((size_name, size_value, image_indices))

Este código se distingue de versiones anteriores por la incorporación de una ventana de Hann para mejorar la reconstrucción de parches y por la paralelización de subíndices de imágenes mediante multiprocessing.Pool. La evaluación de parámetros se realiza con validación cruzada, asegurando que los resultados no dependan de un conjunto de imágenes específico. La estructura modular permite separar claramente el preprocesamiento, la reconstrucción, la evaluación y la optimización, facilitando su mantenimiento y escalabilidad. Los parámetros [n_components, n_iter, alpha, tol] se optimizan mediante GA con elitismo, cruzamiento personalizado y mutación gaussiana, garantizando una exploración eficiente del espacio de soluciones.

El código integra un flujo de procesamiento modular que abarca desde la preparación de imágenes y la segmentación en parches, hasta la reconstrucción con Matching Pursuit y la optimización de parámetros mediante un Algoritmo Genético. La ventana de Hann mejora la calidad de reconstrucción, y la paralelización de subíndices optimiza el tiempo de cómputo. La combinación de GA, validación cruzada y procesamiento paralelo hace que este enfoque sea robusto, eficiente y adecuado para entornos de supercómputo, garantizando resultados reproducibles y confiables para distintos tamaños y subíndices de imágenes.

5.1.4 Código de verificación de hiperparámetros – Checkpoint visual

Algoritmo 4. Evaluación visual de los hiperparámetros obtenidos para MP

Entrada: Carpeta con imágenes {img0.tif, img1.tif, img3.tif, img3.tif}, parámetros del modelo $\langle n\_components, n\_iter, \alpha, tol \rangle$.

Salida: Imagen reconstruida Î para cada imagen de entrada, tamaño en MB de las imágenes originales y las reconstruidas.

1. Para cada imagen $i = 0,1,2,3$ en la carpeta Set#:
    1. Cargar la imagen en escala de grises.
    2. Redimensionar la imagen al tamaño predefinido.
    3. Normalizar los valores de intensidad.
2. Transformar la imagen al dominio frecuencial mediante FFT bidimensional.
3. Aplanar la representación frecuencial y aplicar Dictionary Learning para obtener una representación dispersa.

4. Reconstruir la imagen en el dominio de Fourier a partir del código disperso y el diccionario aprendido.
5. Aplicar la transformada inversa de Fourier para obtener la imagen reconstruida en el dominio espacial.
6. Calcular el tamaño en memoria de la imagen original y la reconstruida.
7. Mostrar visualmente la imagen original y su reconstrucción.

El código mostrado en el Algoritmo 4 fue empleado como punto de control o checkpoint para continuar con el entrenamiento de los diccionarios después de haber obtenido los hiperparámetros óptimos del algoritmo Matching Pursuit mediante un algoritmo genético (GA). Los valores de n_components, alpha, tol y n_iter fueron ajustados previamente por el GA para maximizar la fidelidad de la reconstrucción y optimizar la compacidad de la representación.

El script permite aprovechar directamente esos parámetros ajustados, aplicándolos a todo el conjunto de imágenes sin la necesidad de verificar manualmente los resultados en cada iteración. De esta manera, el flujo de trabajo se automatiza y se asegura la consistencia en el procesamiento masivo de datos, garantizando que todas las imágenes sean tratadas bajo las mismas condiciones experimentales y de optimización. Este enfoque contribuye a la reproducibilidad del experimento y a la eficiencia del entrenamiento, reduciendo significativamente el tiempo requerido para el procesamiento completo del conjunto.

El flujo de trabajo comienza con la carga y redimensionamiento de las imágenes. Cada imagen es redimensionada a un tamaño fijo mediante interpolación LANCZOS, con el propósito de disminuir la carga computacional manteniendo una alta calidad visual. Posteriormente, la imagen se convierte en una matriz de valores normalizados entre cero y uno.

A continuación, la imagen se transforma al dominio de frecuencia mediante la Transformada Rápida de Fourier bidimensional. El espectro resultante es desplazado de modo que las frecuencias bajas se centran, facilitando así su análisis y tratamiento posterior. Esta representación frecuencial es especialmente útil para identificar patrones periódicos o interferencias, características propias de imágenes ópticas y holográficas.

Sobre la representación en el dominio de Fourier se aplica el modelo de aprendizaje de diccionario, el cual busca aprender un conjunto de componentes base o “átomos” que permiten describir la señal de entrada de forma dispersa. El aprendizaje de diccionario ajusta los coeficientes de

combinación de estos átomos de manera que la reconstrucción sea lo más fiel posible al original, pero utilizando una cantidad mínima de componentes, promoviendo así la compacidad de la representación.

Una vez obtenidos los coeficientes dispersos, la imagen se reconstruye multiplicando la representación dispersa por los componentes del diccionario. Posteriormente, se aplica la transformada inversa de Fourier para regresar al dominio espacial. Finalmente, la imagen reconstruida se normaliza y se limita al rango válido de intensidades, asegurando que los valores de píxel estén dentro del intervalo permitido.

El código, a diferencia de los códigos usados para la búsqueda de los hiperparámetros óptimos, incluye una fase de evaluación visual y cuantitativa, en la cual se comparan la imagen original y la reconstruida mostrando sus respectivos tamaños en memoria.

5.1.5 Código para el entrenamiento de los diccionarios de MP

Algoritmo 5. Entrenamiento de los diccionarios de MP

Entrada: Carpeta base con subcarpetas run*, índice de imagen $i = \{0,1,2,3\}$, parámetros del modelo $\langle n\_components, n\_iter, \alpha, tol \rangle$, número máximo de ejemplos N_{ej}.

Salida: Diccionario entrenado D almacenado en un archivo .pkl.

1. Inicializar lista de imágenes normalizadas.
2. Para cada carpeta run_i en la carpeta base:
    1. Cargar la imagen img_k.tif.
    2. Convertir a escala de grises de 8 bits si es necesario.
    3. Redimensionar la imagen al tamaño fijo.
    4. Normalizar intensidades en el rango [0,1].
    5. Almacenar la imagen procesada.
3. Seleccionar aleatoriamente N imágenes si hay más disponibles.
4. Para cada imagen seleccionada:
    1. Extraer parches con tamaño fijo y solapamiento definido.
5. Unir todos los parches en una sola matriz de entrenamiento.
6. Entrenar un diccionario mediante Dictionary Learning usando los hiperparámetros dados.
7. Guardar el diccionario entrenado en un archivo.

El código correspondiente (Algoritmo 5) implementa el proceso de entrenamiento, generación y almacenamiento de diccionarios para representar imágenes mediante la técnica de Dictionary Learning, un método no supervisado que busca aprender una base de átomos (componentes) capaz de reconstruir con alta fidelidad las estructuras locales de una imagen.

El entrenamiento se realiza de forma individual para cada índice de imagen dentro de un conjunto de carpetas estructuradas por ejecuciones (run1, run2, etc.), aplicando parámetros de entrenamiento definidos manualmente. Cada diccionario se guarda en formato binario .pkl con el nombre dictionary_<índice>_<número de ejemplos>.pkl. La estructura interna del archivo contiene una lista con un único arreglo NumPy que almacena los átomos del diccionario aprendidos.

El código está diseñado para mantener coherencia con los métodos de normalización y procesamiento empleados durante la búsqueda de hiperparámetros mediante algoritmos genéticos (GA), garantizando compatibilidad con los procedimientos posteriores de reconstrucción y evaluación visual.

El proceso comienza localizando todas las carpetas con prefijo run dentro de la carpeta base. Para cada carpeta se busca una imagen cuyo nombre siga el patrón img<índice>.tif. Cada imagen es cargada mediante la función load_and_prepare_image, que corrige formatos no estándar y asegura una representación homogénea en escala de grises de 8 bits. Posteriormente, las imágenes son convertidas a matrices NumPy y normalizadas al rango [0,1].

Una vez reunidas todas las imágenes disponibles del mismo índice, se selecciona un subconjunto limitado al número de ejemplos especificado. De cada imagen se extraen parches de 32×32 píxeles con un 50% de solapamiento entre ellos. Los parches de todas las imágenes seleccionadas se concatenan para formar el conjunto de entrenamiento del diccionario.

El entrenamiento se lleva a cabo mediante la función train_dictionary, que instancia el modelo DictionaryLearning de Scikit-learn con los parámetros indicados manualmente. El algoritmo aprende una base de n_components átomos que minimizan el error de reconstrucción sujeto a una penalización de dispersión controlada por el parámetro alpha.

Al finalizar el proceso, los átomos aprendidos se devuelven como una matriz transpuesta y se almacenan dentro de una lista para permitir la compatibilidad con futuras ampliaciones del sistema

que contemplen múltiples diccionarios en un mismo archivo. Esta lista se serializa y guarda en un archivo binario mediante la biblioteca pickle.

El archivo generado recibe el nombre dictionary_<índice>_20.pkl, donde <índice> corresponde al número de imagen con el cual se entrenó el diccionario y el sufijo _20 hace referencia al número de ejemplos utilizados en el entrenamiento.

El contenido del archivo es una lista que contiene un único elemento, el cual es un arreglo de tipo numpy.ndarray. Cada fila del arreglo representa un átomo del diccionario y cada columna corresponde a una de las componentes del vector que define un parche aplanado de la imagen.

5.1.6 Código para la Verificación de los Diccionarios Entrenados – Checkpoint Visual

Algoritmo 6. Checkpoint visual de los diccionarios de MP entrenados

Entrada: Carpeta de imágenes, carpeta con diccionarios, índice de imagen i, tamaño de parche.

Salida: Visualización comparativa entre la imagen original y su reconstrucción con el diccionario correspondiente.

1. Cargar la imagen img_k en escala de grises.
2. Normalizar la imagen al rango [0,1] mediante min–max.
3. Cargar el diccionario correspondiente al índice k.
4. Verificar el formato del diccionario y ajustar dimensiones si es necesario.
5. Dividir la imagen normalizada en parches no solapados de tamaño fijo.
6. Para cada parche:
    1. Calcular su representación dispersa usando Sparse Coding (OMP).
    2. Reconstruir el parche mediante combinación lineal de los átomos del diccionario.
7. Reconstruir la imagen completa a partir de los parches reconstruidos.
8. Normalizar la imagen reconstruida para visualización.
9. Mostrar la imagen original y la reconstruida para comparación visual.

El script presentado en el Algoritmo 6 tiene como propósito evaluar visualmente la calidad de reconstrucción de una imagen a partir de un diccionario previamente entrenado mediante Dictionary Learning. El objetivo es verificar que el diccionario capture adecuadamente las

características de las imágenes con desplazamiento de fase y que los parches reconstruidos reproduzcan de manera fiel la estructura original.

El programa carga una imagen y su correspondiente diccionario, divide la imagen en parches, los codifica en una representación dispersa mediante Sparse Coding y posteriormente los reconstruye para generar una versión aproximada de la imagen original. Finalmente, muestra ambas imágenes (original y reconstruida) lado a lado para comparar los resultados.

El flujo general de trabajo inicia con la función principal visualize_checkpoint, que recibe como parámetros la carpeta donde se encuentra la imagen (set_folder), la carpeta donde está almacenado el diccionario (dict_folder) y el índice de la imagen (img_index). A partir de estos datos, el script ejecuta una secuencia de pasos coherente con los métodos de entrenamiento y normalización empleados durante la generación de los diccionarios.

El primer paso consiste en cargar la imagen mediante load_image, que abre el archivo TIFF correspondiente al índice indicado (img#.tif), lo convierte a escala de grises, lo transforma en un arreglo NumPy de tipo float32 y aplica una normalización min–max, escalando los valores al rango [0,1]. Esta normalización es la misma que se utilizó durante el proceso de optimización genética y entrenamiento de diccionarios, garantizando coherencia entre ambas etapas.

Posteriormente, el script carga el diccionario con la función load_dictionary, que abre el archivo .pkl correspondiente al índice especificado (por ejemplo, dictionary_2_100.pkl). Dado que los diccionarios pueden haberse guardado en diferentes estructuras, el código contempla tres posibles formatos:

- Un arreglo NumPy directamente con los átomos del diccionario.
- Un diccionario de Python con la clave "components_", propio del objeto DictionaryLearning de scikit-learn.
- Una lista que contiene un solo arreglo NumPy (el formato que se genera con el script de entrenamiento, donde cada archivo .pkl guarda una lista con un único diccionario dentro).

En este último caso, el programa extrae el primer elemento de la lista y lo asigna a la variable D. Con ello, el diccionario cargado corresponde a una matriz cuyas columnas representan los átomos o bases aprendidas, y las filas indican el número de componentes que describen cada parche de

imagen. Si el diccionario tuviera una sola fila (caso excepcional), se replica para mantener compatibilidad dimensional con el proceso de reconstrucción.

Una vez cargados la imagen y el diccionario, el programa verifica que sus dimensiones sean compatibles. Si el número de columnas del diccionario no coincide con el tamaño de los parches, el script transpone la matriz. Esto asegura que la estructura final tenga la forma esperada: cada átomo corresponde a una representación vectorial de un parche de imagen aplanado (por ejemplo, 32×32 = 1024 elementos por átomo).

La función reshape_for_dictionary divide la imagen en parches no solapados de 32×32 píxeles, aplanando cada uno en un vector de 1024 valores. Todos los parches se almacenan en una matriz de dimensiones (n_patches, 1024). Este formato coincide con la estructura del diccionario, que posee el mismo número de columnas.

El siguiente paso utiliza SparseCoder de scikit-learn para calcular los coeficientes dispersos (sparse codes) de cada parche en función del diccionario cargado. El algoritmo empleado es Orthogonal Matching Pursuit (OMP), con un máximo de 10 coeficientes no nulos por parche (transform_n_nonzero_coefs=10). El resultado de esta codificación es una matriz de pesos donde cada fila describe la contribución de los átomos del diccionario en la reconstrucción de un parche.

Una vez obtenidos los coeficientes, los parches se reconstruyen multiplicando la matriz de códigos por el diccionario (np.dot(codes, D)), lo que produce una aproximación de los parches originales. Posteriormente, la función reconstruct_from_patches reensambla la imagen completa a partir de estos parches reconstruidos. La función asegura que las regiones superpuestas (si las hubiera) se promedien correctamente para evitar discontinuidades.

La imagen reconstruida se normaliza nuevamente con normalize_image para escalar sus valores a [0,1] y facilitar la comparación visual. Finalmente, el script genera una figura con dos subgráficos: a la izquierda la imagen original y a la derecha la reconstruida. Ambas se muestran en escala de grises y sin ejes para facilitar la evaluación visual de la calidad de reconstrucción.

El diccionario utilizado en este proceso proviene del entrenamiento realizado con el código previo, donde se aplicó DictionaryLearning de scikit-learn sobre parches normalizados. En ese script, el resultado de dict_learning.components_.T se almacenaba dentro de una lista antes de ser guardado con pickle. Por tanto, al cargarlo, el programa obtiene una lista con un único elemento: un arreglo

NumPy de forma (n_features, n_components). Cada columna de esta matriz representa un átomo del diccionario, es decir, un patrón elemental de textura o intensidad aprendido a partir de los parches de las imágenes de entrenamiento.

Este script constituye una herramienta de validación visual de los diccionarios aprendidos mediante Dictionary Learning. Permite comprobar que la reconstrucción de las imágenes a partir de las bases entrenadas conserva la información estructural y de contraste esperada, asegurando que los parámetros optimizados durante la fase de entrenamiento (número de componentes, regularización, iteraciones y tolerancia) producen resultados consistentes con las imágenes originales.

5.1.7 Código para el Preprocesamiento de Imágenes con MP Entrenado

Algoritmo 7. Reconstrucción con Sparse Coding y Cálculo de Métricas para Múltiples Ejecuciones

Entrada: Carpeta de imágenes, carpeta con diccionarios, índice de imagen *i*, carpeta de salida.

Salida: Imágenes reconstruidas por cada ejecución, archivo CSV con métricas por ejecución, promedio global de métricas.

1. Crear la carpeta de salida para almacenar reconstrucciones y métricas.
2. Cargar el diccionario correspondiente al índice de imagen k.
3. Para cada carpeta run# en el conjunto de datos:
4. Cargar la imagen img_k en escala de grises.
5. Normalizar la imagen al rango [0,1].
6. Dividir la imagen en parches no superpuestos.
7. Calcular la representación dispersa de cada parche mediante Sparse Coding (OMP).
8. Reconstruir los parches usando el diccionario entrenado.
9. Reconstruir la imagen completa a partir de los parches.
10. Normalizar y acotar la imagen reconstruida.
11. Calcular las métricas PSNR, MSE y SSIM respecto a la imagen original.
12. Guardar la imagen reconstruida y registrar las métricas en un archivo CSV.
13. Calcular y almacenar el promedio global de PSNR, MSE y SSIM sobre todas las ejecuciones.

El script ubicado en el Algoritmo 7 constituye la etapa de procesamiento masivo y evaluación cuantitativa del sistema de reconstrucción de imágenes basado en Dictionary Learning. Representa el paso posterior a la validación visual de los diccionarios entrenados y tiene como propósito aplicar dichos diccionarios a todas las imágenes disponibles, generar sus reconstrucciones mediante Sparse Coding y calcular métricas objetivas de calidad (PSNR, MSE y SSIM).

El flujo completo implementa un ciclo de procesamiento sistemático sobre las carpetas denominadas run#, que contienen las imágenes experimentales (por ejemplo, img0.tif a img3.tif), comparando las imágenes reconstruidas con sus versiones originales y registrando los resultados en un archivo CSV, junto con las reconstrucciones guardadas en formato tif.

El proceso comienza cargando el diccionario previamente entrenado mediante la función load_dictionary, que abre el archivo correspondiente al índice de imagen especificado (por ejemplo, dictionary_0.pkl). Este archivo fue generado por el script de entrenamiento previo, donde los diccionarios fueron almacenados en formato binario mediante pickle. Cada diccionario consiste en una matriz NumPy cuyas dimensiones representan la relación entre el tamaño del vector de cada parche (por ejemplo, 32×32 = 1024 elementos) y el número de átomos aprendidos (n_components). De esta forma, cada fila de la matriz define un átomo o base que codifica una estructura elemental de textura o intensidad presente en los parches de entrenamiento.

Una vez cargado el diccionario, el script procede a procesar de manera secuencial todas las carpetas run# contenidas en data_folder. En cada carpeta, se busca la imagen correspondiente al índice (img_index) indicado, que es cargada y convertida a un arreglo NumPy de tipo float32. Luego se aplica la función normalize_image, que realiza una normalización min–max de los valores de intensidad al rango [0,1], garantizando coherencia con la normalización utilizada durante la etapa de entrenamiento y con los métodos evolutivos previos.

El siguiente paso es la reconstrucción de la imagen mediante la función reconstruct_image. Este proceso consiste en dividir la imagen normalizada en parches de tamaño cuadrado (por defecto, de 32×32 píxeles), aplanar cada parche en un vector unidimensional y aplicar codificación dispersa (Sparse Coding) usando el objeto SparseCoder de scikit-learn. Se utiliza el algoritmo Orthogonal Matching Pursuit (OMP) con un máximo de 10 coeficientes no nulos por parche (transform_n_nonzero_coefs=10), lo que significa que cada parche es representado como una combinación lineal de hasta 10 átomos del diccionario.

Una vez obtenidos los coeficientes de codificación, la reconstrucción de los parches se realiza mediante la multiplicación matricial entre los coeficientes y el diccionario (np.dot(code, dictionary)), produciendo una aproximación de los parches originales. Los parches reconstruidos se reensamblan luego en la función reconstruct_from_patches, que reconstruye la imagen completa colocando cada bloque en su posición correspondiente y promediando en caso de superposición. La imagen resultante se normaliza nuevamente y se limita al rango [0,1] para evitar valores fuera de escala debido a errores numéricos.

Una vez generada la imagen reconstruida, el código evalúa su calidad mediante la función calculate_metrics, que utiliza las implementaciones de skimage.metrics para calcular tres indicadores:

- PSNR (Peak Signal-to-Noise Ratio): mide la relación entre la señal original y el ruido introducido por la reconstrucción, expresada en decibelios (dB).
- MSE (Mean Squared Error): calcula el error cuadrático medio entre la imagen original y la reconstruida.
- SSIM (Structural Similarity Index): evalúa la similitud estructural percibida, considerando luminancia, contraste y textura.

Estas métricas se calculan para cada conjunto run# y se registran en un archivo CSV (por ejemplo, metrics_img0.csv), junto con el nombre del conjunto y de la imagen procesada. Al final del recorrido, el script calcula el promedio global de las métricas obtenidas en todas las carpetas, lo que permite una evaluación general del rendimiento del diccionario para un determinado índice de imagen.

Además de registrar los valores numéricos, el programa guarda las imágenes reconstruidas en la carpeta de salida definida por output_folder. Cada archivo se guarda en formato tif con un nombre que identifica tanto el conjunto de origen como el índice de imagen (por ejemplo, run5_recon_img0.tif). Estas imágenes permiten una inspección visual complementaria del desempeño del modelo de reconstrucción.

El proceso global implementado por la función process_all_runs automatiza el flujo de reconstrucción y evaluación para múltiples conjuntos de datos, garantizando consistencia en la

comparación entre los resultados. La estructura modular del código facilita su reutilización para distintos índices de imagen o diferentes diccionarios, siempre que se mantenga la coherencia entre el tamaño de los parches y la dimensionalidad del diccionario.

En términos de organización, este script constituye la fase final del ciclo de validación de diccionarios entrenados por Dictionary Learning. Mientras que el código anterior (checkpoint visual) permitía verificar cualitativamente la capacidad del diccionario para reconstruir una imagen individual, este script amplía el análisis a un nivel cuantitativo, sistemático y estadístico. Permite obtener medidas objetivas del desempeño del modelo y consolidar resultados mediante promedios globales que reflejan la estabilidad del método frente a variaciones entre diferentes imágenes experimentales.

Este programa automatiza la evaluación masiva de la reconstrucción basada en Sparse Coding con diccionarios entrenados, generando métricas cuantitativas y salidas visuales que permiten validar la calidad del aprendizaje obtenido y su aplicabilidad general a todo el conjunto de datos experimentales.

5.2. Implementación del Autoencoder

Algoritmo 8. Entrenamiento de Autoencoder mediante Generación de Parches y Validación Cruzada

Entrada: Carpeta de imágenes, tamaño de parche, número de épocas, tamaño de batch *B*, número de folds *K*, parches por run *P*.

Salida: Archivos best_autoencoder_foldX.h5 y autoencoder_final.h5.

1. Definir los parámetros del entrenamiento (tamaño de parches, batch size, épocas y número de folds).
2. Obtener la lista completa de carpetas run# del conjunto de datos.
3. Definir un generador que:
    1. Cargue las imágenes img_0-img_3 de cada run.
    2. Normalice las imágenes.
    3. Extraiga parches aleatorios de tamaño fijo.
    4. Use img_0 como entrada y $\{img_1,img_2,img_3\}$ como salida concatenada.
4. Construir un autoencoder convolucional con:

    1. Un encoder basado en capas convolucionales y max-pooling.
    2. Un decoder basado en convoluciones transpuestas.
    3. Una capa de salida con tres canales.
5. Dividir el conjunto de datos en *K* folds mediante validación cruzada.
6. Para cada fold:
    1. Separar los runs en conjuntos de entrenamiento y validación.
    2. Inicializar el modelo de autoencoder.
    3. Entrenar el modelo usando el generador de parches.
    4. Aplicar early stopping y guardar el mejor modelo según la pérdida de validación.
7. Finalizada la validación cruzada, entrenar un modelo final usando todos los runs.
8. Guardar el modelo entrenado definitivo.

En esta fase, las imágenes reconstruidas obtenidas en pasos previos se emplean como base de datos de entrenamiento para un modelo de autoencoder convolucional (CAE). Su propósito es aprender una representación latente que permita generar automáticamente las imágenes con desplazamiento de fase (img1, img2, img3) a partir de una imagen de referencia (img0).

El uso de este autoencoder permite sustituir el proceso tradicional de reconstrucción basado en MP y DL por un modelo que, una vez entrenado, puede inferir las imágenes reconstruidas de manera directa y mucho más eficiente. Esto reduce significativamente el tiempo de procesamiento y elimina la necesidad de aplicar transformadas de Fourier o ajustes de dispersidad en cada ejecución.

La estructura general del código se divide en tres bloques principales: la preparación de datos mediante generación de parches, la definición del modelo autoencoder, y el entrenamiento supervisado con validación cruzada (Tabla 5.4).

Tabla 5.4 Descripción general de las funciones y bloques principales del Autoencoder

Función / Bloque	***Descripción***	***Propósito***
patch_generator(run_list, batch_size, patches_per_image)	Genera parches aleatorios normalizados desde las imágenes reconstruidas.	Permite un entrenamiento eficiente sin cargar todo el dataset en memoria.
build_autoencoder()	Define la arquitectura del modelo convolucional encoder-decoder.	Aprende representaciones latentes para generar las tres imágenes desplazadas.
KFold(n_splits=5)	Implementa validación cruzada sobre las carpetas de runs.	Evalúa la robustez del modelo y previene sobreajuste.
model.fit(...)	Ejecuta el entrenamiento por cada fold con callbacks de control.	Ajusta los pesos del modelo con early stopping y guarda los mejores resultados.
final_model.fit(...)	Entrenamiento final con todos los datos.	Genera el modelo definitivo para inferencia global.

El primer bloque define los parámetros del modelo, incluyendo las dimensiones de los parches (128x128 píxeles), el tamaño de lote (8), el número de épocas (50) y el número de divisiones para validación cruzada (5). El dataset contiene carpetas denominadas run*, cada una con cuatro imágenes (img0.tif a img3.tif) previamente reconstruidas mediante Matching Pursuit.

El generador de datos, implementado en la función patch_generator, tiene como objetivo optimizar la carga de imágenes y generar dinámicamente los parches de entrada y salida sin necesidad de almacenar un conjunto completo en memoria. Para cada conjunto de datos, la función carga las cuatro imágenes correspondientes, normaliza sus valores dividiéndolos entre 65535 (rango de intensidad de imágenes de 16 bits) y selecciona de manera aleatoria regiones de 128x128 píxeles. Los parches de entrada corresponden a la imagen sin desplazamiento (img0), mientras que la salida esperada está compuesta por la concatenación de las tres imágenes desplazadas (img1, img2, img3), formando un tensor con tres canales. De esta forma, el modelo aprende una correspondencia entre una única entrada y tres salidas simultáneas, que representan distintas fases reconstruidas.

El bloque correspondiente a la construcción del autoencoder se implementa en la función build_autoencoder. La red utiliza una arquitectura simétrica tipo encoder–decoder. El encoder consiste en dos capas convolucionales con activación ReLU y capas de MaxPooling para reducción progresiva de la dimensionalidad, comprimiendo la información espacial en una representación latente compacta. El decoder invierte este proceso mediante capas de Conv2DTranspose, expandiendo la dimensionalidad hasta recuperar la resolución original. La salida posee tres canales, cada uno asociado a una de las imágenes desplazadas. La función de pérdida utilizada es el error cuadrático medio (MSE), acompañado por el error absoluto medio (MAE) como métrica adicional de evaluación.

El tercer bloque implementa la validación cruzada (K-Fold Cross-Validation) con cinco particiones. Esta técnica permite dividir el conjunto total de runs en subconjuntos de entrenamiento y validación, de manera que cada fold evalúe un subconjunto diferente. El objetivo es garantizar que el modelo generalice correctamente ante nuevas imágenes y que su desempeño no dependa de un subconjunto particular de datos.

Durante cada fold, el entrenamiento se ejecuta utilizando los generadores de parches tanto para los datos de entrenamiento como para los de validación. Se emplean callbacks como ModelCheckpoint para almacenar el modelo con menor pérdida de validación y EarlyStopping para detener el entrenamiento si no se observa mejora tras diez épocas consecutivas, evitando así el sobreajuste.

Una vez completada la validación cruzada, el código procede al entrenamiento final utilizando el conjunto completo de datos. Este modelo, entrenado con toda la información disponible, se guarda en los archivos best_autoencoder_final.h5 y autoencoder_final.h5, que representan las versiones optimizada y definitiva del sistema, respectivamente.

El uso de los hiperparámetros obtenidos previamente en las etapas de Matching Pursuit es fundamental, ya que las imágenes reconstruidas que alimentan este modelo derivan directamente de diccionarios entrenados bajo dichos parámetros óptimos. Si se utilizaran reconstrucciones generadas con configuraciones arbitrarias, la calidad y coherencia espacial de las imágenes se vería afectada, impactando negativamente el aprendizaje del autoencoder. Entrenar el modelo sobre datos generados consistentemente garantiza que aprenda las relaciones físicas y estructurales reales entre la imagen base y sus desplazamientos de fase.

5.2.1. Código de Verificación del Modelo – Checkpoint Visual

Algoritmo 9. Evaluación del Autoencoder con 3 Salidas

Entrada: Carpeta de imágenes, modelo entrenado, tamaño de entrada.

Salida: Visualización de las reconstrucciones generadas por el modelo, comparación con las imágenes objetivo reales.

1. Listar las carpetas run# disponibles en el directorio de datos.
2. Seleccionar el run a evaluar.
3. Cargar el modelo de autoencoder entrenado, resolviendo compatibilidades de capas personalizadas.
4. Definir una función de preprocesamiento que:
    1. Redimensione la imagen al tamaño objetivo.
    2. Normalice los valores de intensidad al rango $[0,1]$.
5. Cargar y preprocesar la imagen de referencia img_0.
6. Cargar y preprocesar las imágenes objetivo $\{img_1, img_2, img_3\}$.
7. Introducir la imagen de referencia al modelo entrenado.
8. Obtener las tres imágenes generadas por el autoencoder.
9. Visualizar:
    - Imagen de referencia.
    - Imágenes generadas por el modelo.
    - Imágenes objetivo reales para comparación.
10. Devolver las imágenes generadas.

Tras el entrenamiento del autoencoder con datos generados mediante Matching Pursuit y Dictionary Learning, es fundamental validar si el modelo ha aprendido a reconstruir correctamente las imágenes desplazadas. Con el uso del Algoritmo 9, se puede realizar una inspección visual que permite confirmar la consistencia espacial y tonal de las imágenes generadas, así como detectar posibles artefactos o errores sistemáticos que no se reflejan en las métricas de pérdida.

El script inicia leyendo los directorios disponibles que contienen los conjuntos de imágenes (runs). Luego, solicita al usuario seleccionar el run de interés, asegurando así un control explícito sobre la evaluación. Una vez elegido el conjunto, las imágenes se cargan en memoria, se redimensionan a 128×128 píxeles y se normalizan. El autoencoder entrenado se carga desde su archivo .h5 y se utiliza para generar tres imágenes desplazadas correspondientes a las salidas aprendidas.

Finalmente, las imágenes generadas y las reales se muestran lado a lado en una figura, permitiendo comparar visualmente la precisión de la reconstrucción.

La evaluación visual es una etapa clave dentro del ciclo de validación de modelos generativos, especialmente en contextos donde las métricas numéricas no capturan adecuadamente la percepción visual o la estructura espacial del patrón reconstruido. En este caso, la inspección directa permite comprobar si el autoencoder reproduce de manera coherente las características de fase y contraste que definen las imágenes desplazadas.

Además, la posibilidad de seleccionar manualmente el run a evaluar facilita un análisis más detallado, permitiendo estudiar el comportamiento del modelo sobre casos específicos, por ejemplo, condiciones experimentales distintas o variaciones de ruido en las imágenes.

Al ejecutar el script, se espera que el modelo genere tres imágenes sintéticas correspondientes a los desplazamientos de fase aprendidos. Si el entrenamiento ha sido exitoso, las imágenes resultantes deben presentar una correspondencia visual clara con las imágenes objetivo, tanto en la estructura general como en los patrones de intensidad. Cualquier desviación perceptible puede indicar la necesidad de ajustar hiperparámetros, incrementar la cantidad de datos o modificar la arquitectura del modelo.

5.3. Recuperación de Fase

Algoritmo 10. Cálculo de fase por 4 pasos y evaluación cuantitativa

Entrada: Carpeta de imágenes originales, carpeta de reconstrucciones, número de runs a procesar, directorio de salida.

Salida: Fases originales y reconstruidas en .npy y .tif, CSV con métricas por run, archivo con estadísticas globales.

1. Para cada run *r* del conjunto de datos:
    1. Cargar las imágenes originales $\{img_0, img_1, img_2, img_3\}$.
    2. Cargar las imágenes reconstruidas $\{img_1, img_2, img_3\}$ y usar img_0 como referencia.
    3. Calcular la fase original φ_{orig} mediante el método de cuatro pasos.
    4. Calcular la fase reconstruida φ_{recon} mediante el mismo método.
    5. Evaluar la similitud entre φ_{orig} y φ_{recon} usando:

- PSNR
- SSIM
- RMSE
- Correlación lineal

6. Guardar las fases en formato numérico sin normalizar.
7. Guardar versiones normalizadas de las fases únicamente para visualización.
8. Almacenar las métricas del run en un archivo CSV.

2. Calcular las métricas promedio globales sobre todos los runs procesados.
3. Guardar las estadísticas globales en un archivo de texto.
4. Mostrar en pantalla los valores promedio finales.

En técnicas de interferometría y holografía con luz no detectada, la información de fase es un elemento esencial que contiene los detalles estructurales del frente de onda. Sin embargo, la fase no puede ser medida directamente, por lo que se emplean métodos indirectos, como los algoritmos de phase shifting, para recuperarla a partir de mediciones de intensidad.

El método de cuatro pasos es uno de los más utilizados por su simplicidad y precisión, y su expresión analítica permite calcular la fase de cada píxel con una relación trigonométrica. La recuperación de fase es una etapa crítica para evaluar la calidad de las imágenes reconstruidas. Aplicar este método tanto sobre las reconstrucciones de Matching Pursuit como sobre las generadas por el autoencoder permite:

- Verificar la consistencia física de las imágenes reconstruidas.
- Evaluar si el modelo de aprendizaje (autoencoder) conserva la información interferométrica de la fase.
- Validar el poder reconstructivo de los diccionarios entrenados, al comprobar que las imágenes procesadas por MP conservan la coherencia de fase esperada.

Por tanto, este procedimiento constituye una herramienta de validación experimental y teórica en la cadena de procesamiento holográfico.

El código del Algoritmo 9 implementa un proceso completo que inicia con la carga de datos y finaliza con la generación de métricas estadísticas globales, con el propósito de analizar de manera sistemática el desempeño del método de reconstrucción empleado.

El sistema comienza con la carga de las imágenes TIFF de intensidad que componen cada experimento. Para garantizar uniformidad en el procesamiento, cada archivo se convierte explícitamente al modo de escala de grises y se almacena como un arreglo numérico en formato de punto flotante.

Una vez cargadas las imágenes, se procede al cálculo de la fase aplicando el método de cuatro pasos. Este procedimiento utiliza las imágenes con desplazamientos de fase conocidos y aplica la relación arcotangente correspondiente para obtener la fase en cada punto del plano. Tanto la fase original como la fase reconstruida se calculan directamente en radianes, sin ningún tipo de normalización para mantener su integridad matemática. Únicamente para efectos de visualización se incluye un proceso opcional de normalización lineal que permite exportar versiones TIFF de las fases, facilitando la inspección visual sin alterar la información utilizada en el análisis cuantitativo.

El sistema incorpora posteriormente un conjunto de métricas diseñadas para evaluar la similitud entre la fase original y la fase obtenida a partir de las imágenes reconstruidas. Estas métricas incluyen el PSNR para estimar la degradación de señal, el SSIM para identificar cambios estructurales y patrones locales, el RMSE para cuantificar el error absoluto promedio y la correlación de Pearson para medir la correspondencia global entre ambas fases. La combinación de estas métricas permite evaluar el desempeño tanto en términos locales como globales, proporcionando una caracterización completa de la calidad de la reconstrucción.

El flujo de procesamiento está organizado de manera modular a través de una función dedicada a manejar cada experimento individual. Esta función se encarga de identificar las carpetas asociadas a cada run, cargar las imágenes correspondientes, calcular la fase original y reconstruida, obtener las métricas y generar los archivos de salida. Además de los valores numéricos, se guardan copias de las fases en formato npy sin normalizar, lo que permite realizar análisis posteriores sin pérdida de precisión. De manera paralela se almacenan versiones TIFF normalizadas que constituyen únicamente una representación visual de apoyo.

El sistema cuenta también con un módulo que automatiza el procesamiento de un número definido de runs. Este módulo crea un archivo CSV para almacenar los resultados individuales y ejecuta de forma secuencial el procesamiento de cada experimento. Los valores obtenidos se acumulan para permitir el cálculo de estadísticas globales al finalizar el análisis, lo que facilita la comparación entre distintos métodos de reconstrucción o distintas condiciones experimentales. Estas

estadísticas representan promedios globales para cada una de las métricas calculadas y se escriben en un archivo de texto que actúa como resumen del comportamiento general del sistema.

Finalmente, el sistema establece un conjunto de rutas hacia las carpetas donde se encuentran almacenadas tanto las imágenes originales como las reconstrucciones y define la carpeta de salida donde serán depositados los resultados. Una única instrucción ejecuta por completo el análisis sobre la cantidad especificada de runs y produce todos los archivos necesarios para la evaluación detallada del método. La estructura resultante constituye un flujo robusto y reproducible, adecuado para estudios experimentales donde se requieren análisis cuantitativos precisos sobre grandes volúmenes de datos.

CAPÍTULO 6

ANÁLISIS Y DISCUSIÓN DE LOS RESULTADOS

6.1. Análisis Comparativo entre Métodos Utilizados

En esta sección se presenta un análisis comparativo de las tres versiones del código de optimización de Matching Pursuit mediante Algoritmo Genético desarrolladas a lo largo del proyecto. Cada versión fue implementada con diferentes objetivos y restricciones, desde la primera versión básica con procesamiento secuencial hasta las versiones posteriores adaptadas para compatibilidad con Python 3.6.8 y optimización en entornos de supercómputo. Se destacan las diferencias en la arquitectura, el manejo de imágenes, la aplicación de ventanas de Hann, la validación cruzada, la paralelización y la exploración de parámetros mediante el Algoritmo Genético. Esta comparación permite identificar las mejoras técnicas implementadas en cada versión y su impacto en la eficiencia computacional y la calidad de la reconstrucción de imágenes.

La primera versión constituye la implementación base del algoritmo, caracterizada por un procesamiento completamente secuencial, sin validación cruzada ni aplicación de ventana de Hann, lo que limita tanto su robustez como la calidad de la reconstrucción de los parches. La segunda versión adapta el código para asegurar compatibilidad con Python 3.6.8, incorporando la optimización del parámetro tol y ajustes en DictionaryLearning que garantizan mayor estabilidad, aunque mantiene el procesamiento secuencial y carece de suavizado de bordes.

En contraste, la tercera versión introduce mejoras significativas que optimizan la eficiencia y la precisión. Entre ellas se incluyen la paralelización por subíndices, la aplicación de la ventana de Hann para reducir efectos de borde, la validación cruzada para garantizar la robustez de los parámetros y el soporte para distintos tamaños de imagen, conservando la modularidad de la arquitectura. Esta evolución progresiva de las versiones evidencia un aumento claro en robustez, eficiencia y calidad de reconstrucción, posicionando la tercera versión como la más completa y adecuada para entornos de supercómputo y conjuntos de datos extensos.

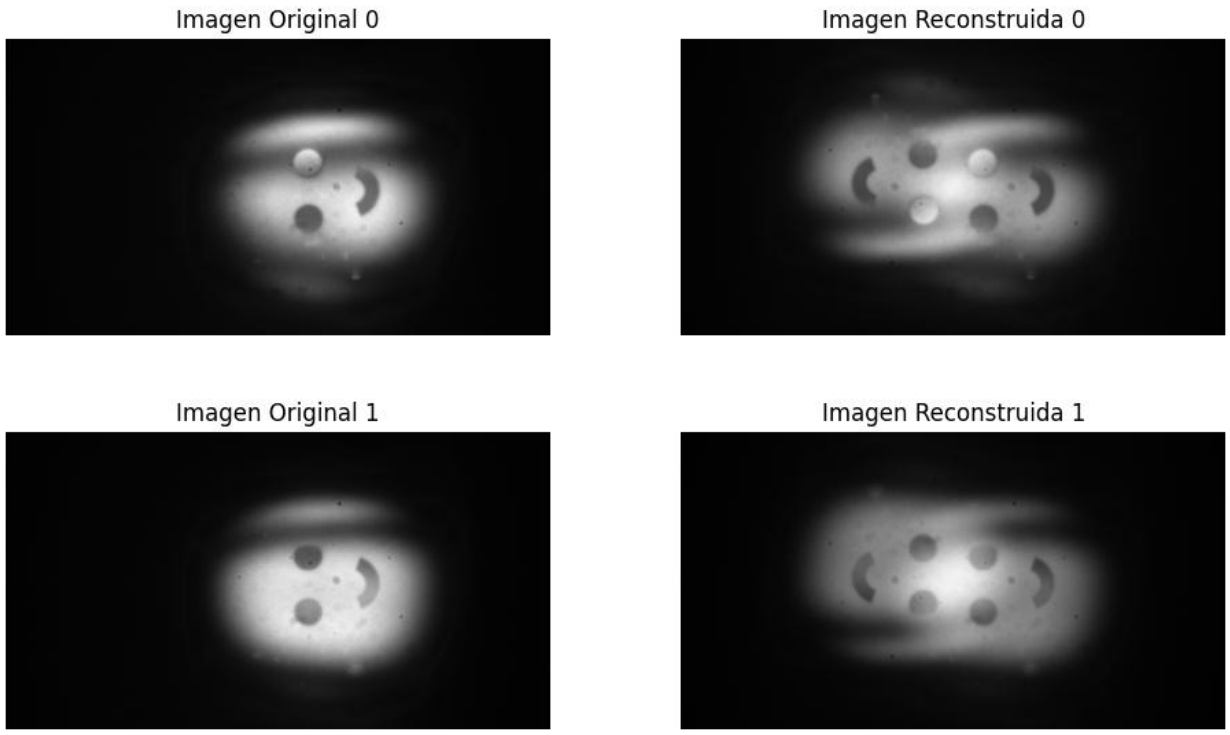
Tabla 56.1 Comparación de las tres versiones del código de Matching Pursuit optimizado con Algoritmo Genético, destacando diferencias en arquitectura, preprocesamiento, paralelización y optimización de parámetros.

Aspecto / Versión	Versión 1	Versión 2	Versión 3
Lenguaje / Compatibilidad	Python 3.7+	Python 3.6.8 (adaptada)	Python 3.6.8 (adaptada con mejoras de paralelización)
Carga de imágenes	Solo escala de grises, normalización entre 0 y 1	Igual que V1, incluye redimensionamiento opcional	Igual que V2, permite distintos tamaños (original, mediana, pequeña)
Preprocesamiento	División en parches de 32×32 con 50% solapamiento	Igual que V1	Igual que V2, pero agrega ventana de Hann 2D aplicada a cada parche
Transformación de parches	FFT y centrado de frecuencia (fftshift)	Igual que V1	Igual que V2
Dictionary Learning / Matching Pursuit	DictionaryLearning con fit_algorithm='lars', transform_algorithm='omp'	Cambiado a lasso_lars para compatibilidad con Python 3.6.8; limitado número de átomos	Igual que V2, con límite de n_components ajustable según tamaño del parche
Reconstrucción de imagen	Promedio de parches solapados	Igual que V1	Igual que V2, incluye prevención de división por cero (np.maximum(count,1e-8))
Evaluación / Métrica	Error cuadrático medio (MSE)	Igual que V1	Igual que V2, con validación cruzada incorporada por pliegues
Optimización de parámetros (GA)	Individuos [n_components, n_iter, alpha]	Añade tol como parámetro, operadores de selección, mutación gaussiana, cruzamiento con cxBlend	Igual que V2, pero cruzamiento personalizado custom_mate para mejorar exploración de parámetros
Paralelización	No implementada	No implementada, todo secuencial	Se paralelizan los subíndices de imágenes dentro de cada tamaño usando multiprocessing.Pool
Validación cruzada	No implementada	No implementada	Implementada por pliegues para asegurar robustez de los parámetros
Ventana de Hann	No	No	Aplicada a cada parche antes de MP para suavizar bordes y mejorar reconstrucción
Manejo de distintos tamaños	Solo tamaño fijo	Posibilidad de redimensionar	Soporta múltiples tamaños, procesando cada tamaño secuencialmente y subíndices en paralelo
Almacenamiento de resultados	Impresión por pantalla	Guardado en archivo .pkl	Guardado en archivo .pkl por tamaño y subíndice, con diccionario completo de parámetros óptimos

6.2. Interpretación de los Resultados

6.2.1 Resultados de la Búsqueda Evolutiva

El proceso de optimización comenzó con una fase preliminar en la que se implementó una versión simplificada del algoritmo genético (GA). Esta primera aproximación consideró un número reducido de hiperparámetros y tuvo como objetivo evaluar la viabilidad del uso de técnicas evolutivas para ajustar el comportamiento del algoritmo de Matching Pursuit (MP). Esta versión preliminar permitió identificar rangos razonables para los parámetros clave y, además, evidenció la necesidad de utilizar una evaluación más robusta para evitar convergencias erróneas o soluciones inestables (Figura 6.1).



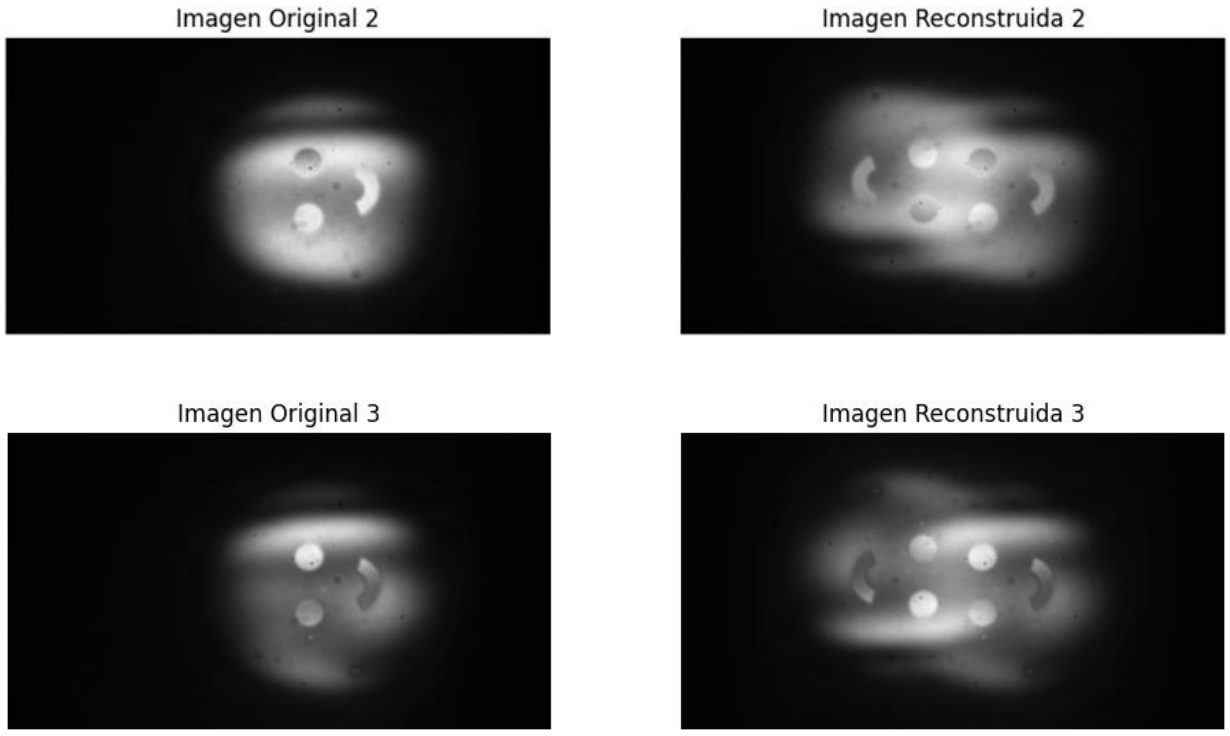


Figura 6.1 Resultados preliminares del algoritmo Matching Pursuit aplicados a los patrones de interferencia. Se observa la calidad inicial de las reconstrucciones obtenidas antes del ajuste fino de hiperparametros

A partir de estas observaciones, se desarrolló una segunda versión del GA, esta vez más completa y robusta, incorporando una estrategia de validación cruzada durante la búsqueda. Esta mejora permitió evaluar cada conjunto de parámetros sobre múltiples muestras, asegurando que los valores óptimos no fueran producto de una imagen atípica o de ruido específico. El GA robusto fue el que finalmente se utilizó para la búsqueda intensiva de parámetros óptimos en los tres desplazamientos de fase.

Inicialmente se consideraron tres escalas de imagen: pequeña, mediana y completa. Sin embargo, tanto el tamaño mediano como el completo presentaron tiempos de ejecución excesivos, superando los 11 días incluso con paralelización multiproceso. Por esta razón, la búsqueda exhaustiva se realizó únicamente con las imágenes reducidas (“tamaño pequeño”) y considerando un desplazamiento a la vez, lo que permitió completar el proceso en un tiempo adecuado y con una evolución estable del algoritmo.

El GA robusto finalizó la búsqueda produciendo los siguientes parámetros óptimos:

- Desplazamiento 1:
  n_components = 263, n_iter = 84, alpha = 0.6193, tol = 0.1881
- Desplazamiento 2:
  n_components = 90, n_iter = 75, alpha = 0.7104, tol = 0.1461
- Desplazamiento 3:
  n_components = 252, n_iter = 74, alpha = 0.8398, tol = 0.1757

Los resultados muestran que los desplazamientos 1 y 3 requieren diccionarios más extensos (más de 250 átomos), mientras que el desplazamiento 2 converge hacia una estructura más compacta. Esta diferencia sugiere que los patrones interferométricos asociados a cada desplazamiento presentan niveles distintos de complejidad estructural, lo que se refleja en las capacidades de representación necesarias.

La inclusión de validación cruzada en la evaluación del GA contribuyó a que los parámetros seleccionados se mantuvieran estables y consistentes a lo largo de las generaciones, evitando sobreajuste dentro de la búsqueda misma.

6.2.2 Integración en los Checkpoints Visuales

Los parámetros obtenidos se integraron en un primer checkpoint visual para evaluar cualitativamente la efectividad de los diccionarios entrenados con dichos valores. En esta fase inicial se observaron artefactos en forma de reflejos residuales, los cuales indicaban que el diccionario aún no capturaba plenamente la estructura característica de los datos (Figura 6.2).

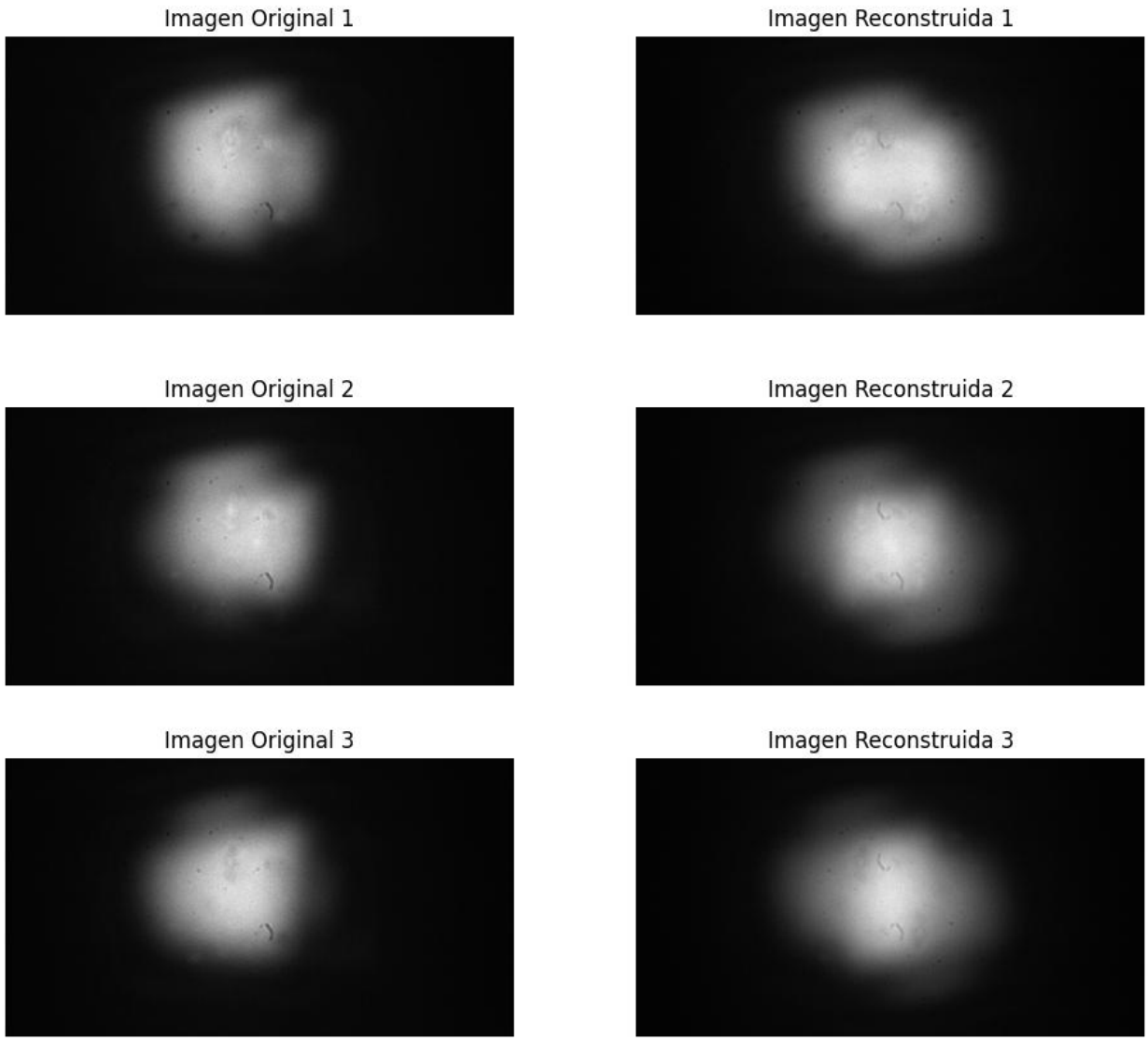


Figura 6.2 Reconstrucciones generadas por Matching Pursuit utilizando los hiperparametros optimizados mediante el proceso de selección y validación cruzada. Las imágenes muestran la mejora obtenida respecto a las versiones preliminares.

Para mejorar la estabilidad del modelo, se integró un entrenamiento de los diccionarios, utilizando 20 ejemplos por desplazamiento. Esto permitió que los diccionarios aprendidos representaran de manera más precisa la variabilidad de los patrones interferométricos. En el segundo checkpoint visual ya no se observaron reflejos ni artefactos importantes, lo cual confirmó que el proceso de entrenamiento del diccionario había alcanzado un punto de consolidación (Figura 6.3).

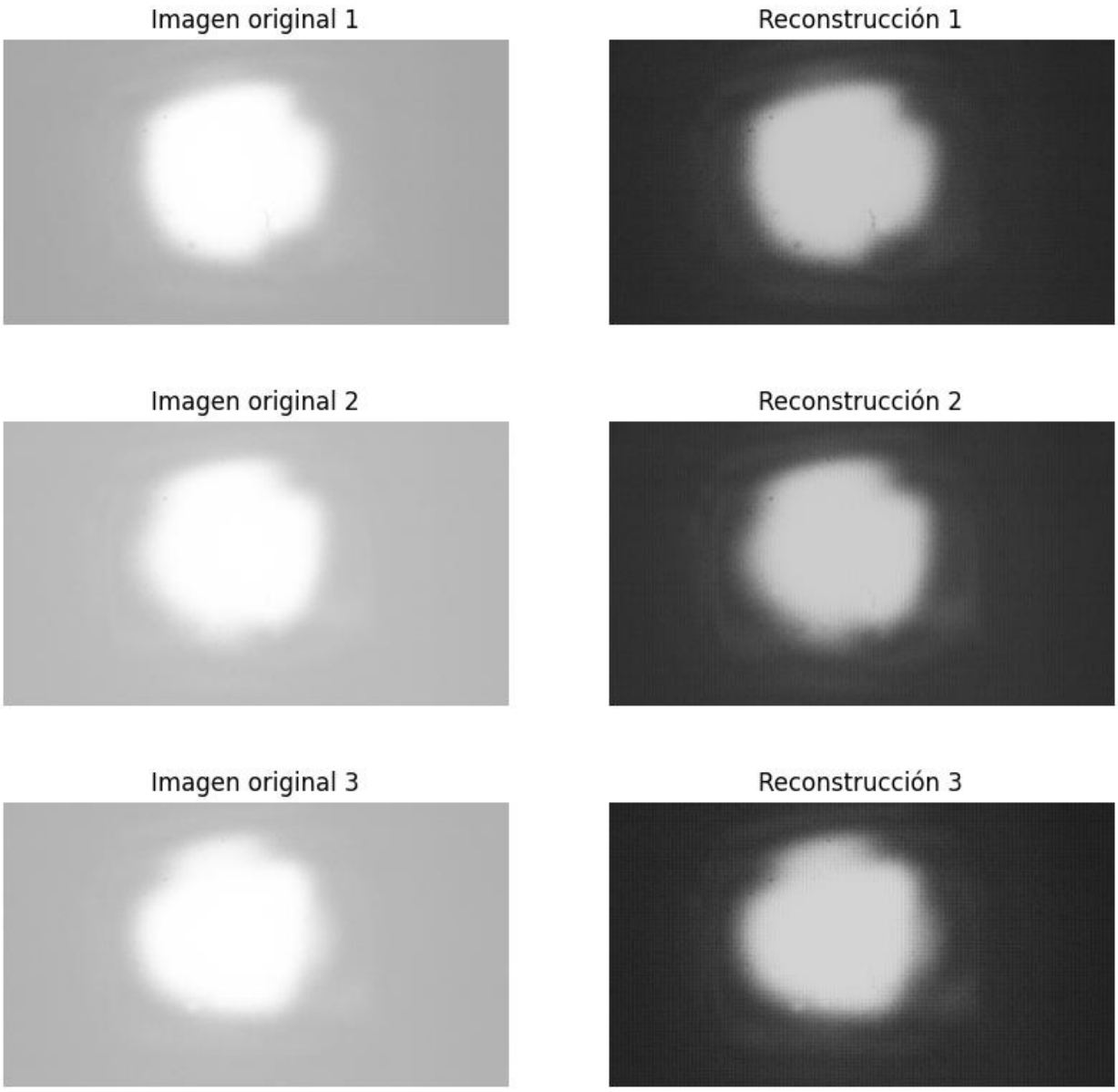


Figura 6.3 Reconstrucciones finales obtenidas tras entrenar los diccionarios específicos para los conjuntos de imágenes.

6.2.3 Reconstrucciones Finales y Evaluación Cuantitativa

Una vez validados visualmente, los diccionarios entrenados con los parámetros optimizados mediante el algoritmo genético se utilizaron para reconstruir las 100 imágenes correspondientes a cada desplazamiento. Para evaluar la calidad de estas reconstrucciones se consideraron cuatro

métricas cuantitativas: PSNR, SSIM, RMSE y Correlación (Corr) entre la imagen original y la reconstruida.

Con el fin de analizar el impacto del tamaño del conjunto de entrenamiento del diccionario, se entrenaron dos versiones del aprendizaje del diccionario:

- Diccionarios entrenados con 20 ejemplos por desplazamiento
- Diccionarios entrenados con 100 ejemplos por desplazamiento

6.2.3.1 Diccionarios entrenados con 20 ejemplos

Los resultados mostraron una calidad de reconstrucción estable, con valores altos de correlación y RMSE bajos (Tabla 6.2.1).En esta configuración se observó, consistentemente, un mejor desempeño del desplazamiento 3, el cual obtuvo los valores más altos de SSIM y Corr, además del RMSE más bajo. Esto coincide con los resultados del GA, donde este desplazamiento requería un diccionario más grande, posiblemente porque posee mayor variabilidad estructural que pudo ser capturada adecuadamente en el entrenamiento.

Tabla 6.2 Promedios asociados a las métricas calculadas para el conjunto de imágenes procesadas con el diccionario entrenado considerando 20 ejemplos.

Desplazamiento	*PSNR*	*SSIM*	*RMSE*	*Corr*
1	25.6386	0.3792	0.0529	0.8827
2	25.3640	0.3613	0.0545	0.8840
3	25.6814	0.3942	0.0526	0.8876

6.2.3.2 Diccionarios entrenados con 100 ejemplos

Al aumentar el número de ejemplos utilizados para entrenar el diccionario, las métricas mostraron un comportamiento distinto (Tabla 6.2.2).

Tabla 76.3 Promedios asociados a las métricas calculadas para el conjunto de imágenes procesadas con el diccionario entrenado considerando 100 ejemplos.

Desplazamiento	*PSNR*	*SSIM*	*RMSE*	*Corr*
1	25.4115	0.3680	0.0542	0.8760
2	25.1444	0.3464	0.0558	0.8776
3	25.3529	0.3745	0.0545	0.8782

Aunque los valores siguen siendo consistentes, los resultados muestran una ligera disminución general en PSNR, SSIM y Corr en comparación con el entrenamiento con 20 ejemplos. Este comportamiento sugiere que, en este caso específico, un conjunto de entrenamiento demasiado amplio puede introducir mayor variabilidad en los parches aprendidos, reduciendo la capacidad del diccionario para representar de manera óptima los patrones característicos del conjunto de imágenes analizado.

Esto puede atribuirse a que las imágenes de interferencia empleadas presentan características altamente repetitivas; por lo tanto, un diccionario entrenado con un conjunto más concentrado de ejemplos facilita que los átomos aprendidos se especialicen en las estructuras más frecuentes.

Los dos escenarios permiten concluir que:

- Ambos tamaños de entrenamiento producen reconstrucciones coherentes y estables.
- Los diccionarios entrenados con 20 ejemplos ofrecen un rendimiento ligeramente superior en todas las métricas.
- La métrica Corr confirma un nivel alto de similitud estructural, con valores entre 0.876 y 0.887 según el caso.
- No se observan diferencias abruptas entre desplazamientos, lo que indica que el proceso de reconstrucción es robusto frente a variaciones en el estado de fase.
- El desplazamiento 3 se mantiene como el que produce las reconstrucciones más fieles en ambos escenarios.

En conjunto, estas evaluaciones cuantitativas verifican que el pipeline implementado —optimización vía GA, aprendizaje de diccionarios y reconstrucción con MP— es estable y reproduce adecuadamente la información relevante de las imágenes interferométricas.

6.2.4 Entrenamiento del Autoencoder

6.2.4.1 Entrenamiento con imágenes sin procesar

Debido al tiempo de cómputo requerido por el preprocesamiento basado en Matching Pursuit (MP), se decidió comenzar la configuración y entrenamiento del autoencoder utilizando directamente las imágenes sin reconstruir. Estas primeras pruebas tuvieron como objetivo determinar si el modelo era capaz de aprender, incluso de manera parcial, la estructura asociada a los desplazamientos de fase antes de contar con el conjunto completo de imágenes procesadas. Sin embargo, al disponerse únicamente de cien imágenes por desplazamiento, fue necesario recurrir a la generación de parches como estrategia de aumento de datos. Se generaron versiones del entrenamiento utilizando mil y diez mil parches por imagen, incorporando además un esquema de validación cruzada en cinco subconjuntos. Durante el proceso se almacenaron los mejores modelos por fold y un modelo general, aunque no fue posible recuperar los archivos estadísticos debido a su tamaño superior a dos gigabytes.

En la configuración de mil parches por imagen, los resultados obtenidos en las predicciones fueron muy deficientes. Las imágenes generadas aparecieron completamente negras, con un patrón cuadriculado evidente y una única mancha oscura concentrada en la región central. Esta tendencia señala que el autoencoder no logró capturar la estructura interferométrica subyacente, limitándose a aprender únicamente la periodicidad introducida por el muestreo en parches (Figura 6.4).

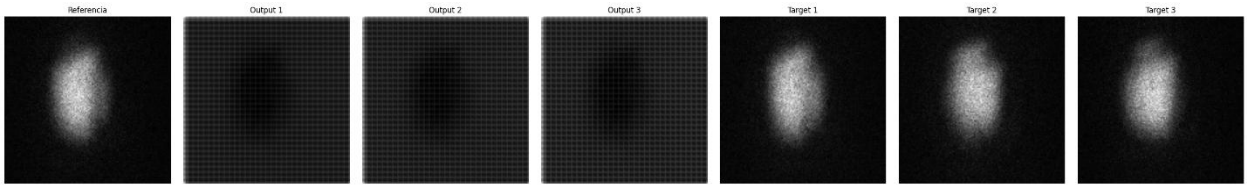


Figura 6.4 Resultados intermedios del autoencoder entrenado con un conjunto de 1000 parches. Se aprecia que las reconstrucciones presentan patrones cuadriculados y regiones oscurecidas, característicos de un modelo subentrenado.

En el caso del entrenamiento con diez mil parches por imagen, se observó un avance moderado. Las imágenes producidas dejaban de ser totalmente negras y comenzaban a mostrar cuadros en tonos de gris claro, con una zona central más oscura. Aunque el modelo empezó a identificar la región de interés asociada al desplazamiento, la información reconstruida seguía siendo fragmentada y dominada por el patrón cuadriculado característico del método de muestreo (Figura 6.5).

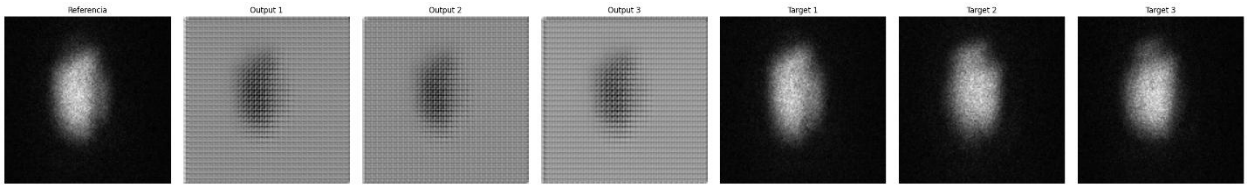


Figura 6.5 Resultados intermedios del autoencoder entrenado con 10000 parches. Las reconstrucciones muestran una estructura más definida y una reducción del cuadriculado observado en el caso con 1000 parches.

6.2.4.2 Entrenamiento del autoencoder con imágenes reconstruidas mediante Matching Pursuit

Una vez completado el preprocesamiento mediante MP fue posible realizar un entrenamiento más robusto del autoencoder utilizando datos significativamente más informativos. Las imágenes reconstruidas representan una versión limpia y depurada de los patrones de interferencia, lo que permite que la red aprenda directamente estructuras vinculadas al desplazamiento y no artefactos derivados del ruido o del patrón interferométrico original.

El nuevo entrenamiento se configuró empleando parches de 128×128 píxeles, un canal, un tamaño de lote de ocho, treinta épocas de entrenamiento, validación cruzada con tres subconjuntos y un total de diez mil parches por imagen. Esta configuración permitió un equilibrio adecuado entre capacidad de generalización y costo computacional.

Con el objetivo de evaluar el comportamiento del modelo, se realizó un checkpoint visual utilizando un conjunto de prueba externo al entrenamiento. En esta evaluación, las imágenes generadas presentaron una mejora notable respecto a las obtenidas en los entrenamientos

preliminares. Aunque la cuadrícula derivada del muestreo en parches seguía siendo visible, las predicciones mostraron un patrón predominante blanco, con una región central negra claramente delimitada. Este comportamiento indica que el autoencoder logró reconocer y reproducir la zona donde se encuentra la mayor variación espacial del desplazamiento, reflejando un aprendizaje parcial coherente con la información disponible en las imágenes procesadas por MP (Figura 6.6).

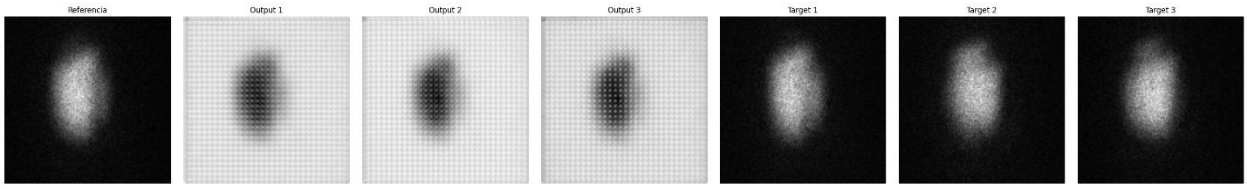


Figura 6.6 Reconstrucciones generadas por el autoencoder entrenado con 10000 parches, utilizando como entrada las imágenes reconstruidas mediante Matching Pursuit. Se observa una mayor homogeneidad en la intensidad y una mejora en la recuperación de los detalles centrales.

6.2.5 Evaluación Mediante el Cálculo de Fase

Como última etapa del análisis, se realizó el cálculo de fase tanto para las imágenes originales como para las imágenes reconstruidas mediante el algoritmo Matching Pursuit. El objetivo de este procedimiento era comparar directamente la fidelidad de las reconstrucciones en términos de la información de fase, que corresponde al parámetro físicamente relevante asociado a los desplazamientos utilizados en el experimento. La misma evaluación estaba prevista para las imágenes generadas por el autoencoder; sin embargo, tras la inspección visual se determinó que las predicciones no alcanzaban la resolución espacial necesaria para obtener una fase coherente. Debido a ello, las imágenes del autoencoder no fueron consideradas en esta última etapa de evaluación cuantitativa.

Para las imágenes reconstruidas a través de MP, se calcularon las mismas métricas utilizadas en el análisis previo, comparando ahora la fase recuperada con la fase original. Esta evaluación reveló que, aunque las reconstrucciones contienen suficiente información para reproducir parcialmente la estructura interferométrica, la fidelidad en términos de fase sigue siendo limitada. Los valores promedio obtenidos en esta evaluación fueron los siguientes: PSNR = 10.3332, SSIM = 0.0150, RMSE = 1.9129 y CORR = 0.0852. Estos resultados reflejan que la relación entre la fase original

y la fase derivada de las imágenes reconstruidas es débil, lo que indica una pérdida considerable de información en esta dimensión específica.

El bajo valor de correlación confirma que la estructura fina de la fase no se preserva adecuadamente, aún cuando la reconstrucción en amplitud, evaluada previamente con las métricas tradicionales, presenta valores aceptables en términos de similitud global. Esta diferencia evidencia que la recuperación de fase es un problema mucho más sensible a pequeñas variaciones locales en la imagen, las cuales pueden quedar enmascaradas en métricas basadas exclusivamente en intensidad. En este sentido, el Matching Pursuit permitió una reconstrucción visualmente coherente y estructuralmente comparable, pero no logró preservar con precisión la información de fase de los patrones interferométricos originales (Figura 6.7).

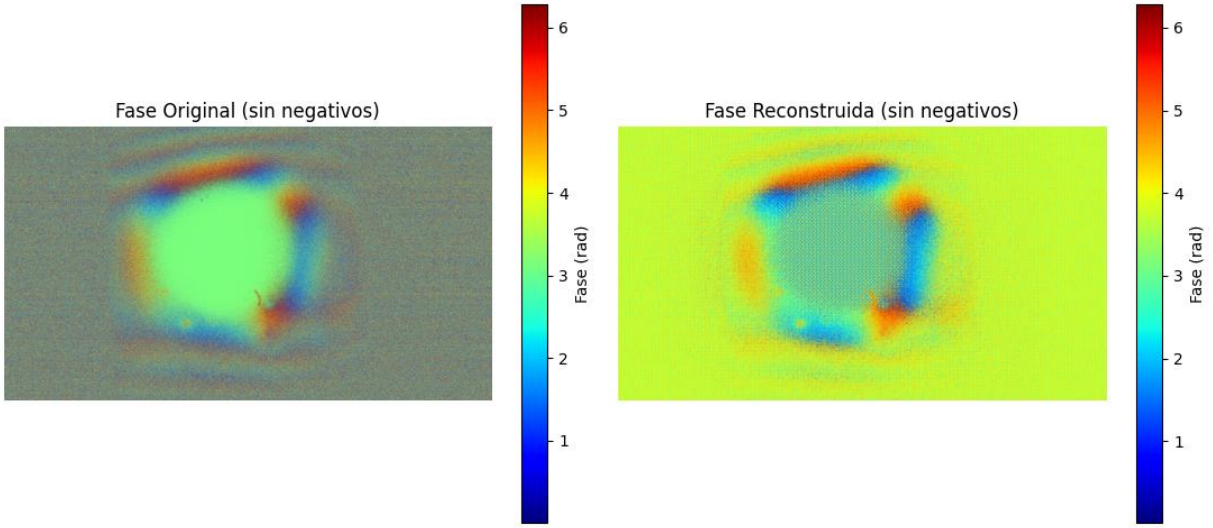


Figura 6.7 Comparación entre la fase original y la fase reconstruida a partir de los desplazamientos de fase. La fase recuperada reproduce correctamente las variaciones principales, aunque presenta presencia de bloques debido al proceso de reconstrucción.

Dado que las predicciones del autoencoder no alcanzaron un nivel de calidad suficiente para conservar la estructura interferométrica en intensidad, era de esperarse que su evaluación en términos de fase resultara aún menos informativa. Esto justifica su exclusión en esta etapa del análisis, aunque también subraya la necesidad de explorar arquitecturas más apropiadas o estrategias de entrenamiento basadas en la fase directamente como objetivo de aprendizaje.

6.3. Discusión sobre las Limitaciones del Método

El desarrollo del sistema de reconstrucción permitió identificar varias limitaciones inherentes tanto al uso de Matching Pursuit (MP) como a los modelos de autoencoder empleados. En primer lugar, el proceso de búsqueda evolutiva de hiperparámetros, aun cuando permitió obtener configuraciones óptimas para el entrenamiento de diccionarios, resultó altamente costoso en términos computacionales. Las ejecuciones para tamaños mediano y completo no lograron finalizar incluso utilizando multiprocesamiento, con tiempos estimados superiores a once días. Esto obligó a restringir la búsqueda al tamaño reducido, limitando la generalización del método a otras resoluciones.

Una segunda limitación proviene del propio MP. Aunque las imágenes reconstruidas permiten recuperar estructuras espaciales principales y eliminar artefactos como el reflejo, el algoritmo no preserva fielmente la información de fase. Esto quedó evidenciado en los valores promedio obtenidos al comparar la fase reconstruida con la original, donde se obtuvo PSNR ≈ 10.3, SSIM ≈ 0.015, CORR ≈ 0.085 y RMSE elevado. Estos resultados muestran que, aun cuando las reconstrucciones presentan similitud visual cualitativa, no retienen adecuadamente la información interferométrica fina ni la continuidad requerida para una recuperación precisa de fase.

El autoencoder, por su parte, mostró limitaciones aún más pronunciadas. Los entrenamientos realizados con imágenes sin procesar no lograron aprender la estructura interferométrica, produciendo salidas dominadas por patrones cuadriculados derivados del muestreo por parches. Incluso al aumentar el número de parches a diez mil por imagen, el modelo únicamente recuperó zonas oscuras o claras en regiones centrales, sin aproximarse a las estructuras reales de los desplazamientos. La falta de continuidad espacial y la dependencia total del mosaico de parches redujeron drásticamente la calidad de la señal generada.

Finalmente, incluso el entrenamiento utilizando imágenes reconstruidas con MP mantuvo el problema de la estructura cuadriculada, ya que el autoencoder continúa aprendiendo patrones locales en lugar de distribuciones globales. Si bien este entrenamiento logró recuperar un área central negra que corresponde al desplazamiento, las predicciones carecieron de detalle, amplitud

dinámica y resolución; por este motivo, no pudieron ser utilizadas para obtener mapas de fase. Esta incapacidad del modelo para mantener coherencia espacial global representa una limitación crítica del enfoque.

6.4. Posibles Mejoras y Trabajo Futuro

A partir del análisis de resultados y las limitaciones identificadas, existen múltiples direcciones prometedoras para mejorar el desempeño de los métodos de reconstrucción.

Un primer aspecto consiste en optimizar el proceso de búsqueda evolutiva. La introducción de métodos híbridos que reduzcan el espacio de búsqueda puede disminuir drásticamente los tiempos de ejecución. A pesar de que los experimentos se realizaron en un entorno de supercómputo, el proceso de búsqueda evolutiva continuó siendo altamente demandante. Una mejora natural consiste en aprovechar nodos especializados dentro del mismo sistema, particularmente aquellos que cuentan con aceleradores GPU o configuraciones multi-GPU, lo cual permitiría evaluar tamaños de imagen mayores y reducir significativamente los tiempos de ejecución. Asimismo, la optimización del paralelismo mediante MPI, OpenMP o versiones distribuidas del algoritmo podría mejorar el uso de los recursos disponibles y acelerar la búsqueda de hiperparámetros sin requerir cambios sustanciales en la arquitectura del sistema.

Respecto al MP, sería conveniente explorar variantes más robustas o métodos alternativos al sparse coding tradicional, tales como Orthogonal Matching Pursuit mejorado, K-SVD con términos de regularización específicos, o modelos que integren información espectral y espacial simultáneamente. Otra línea consiste en incorporar directamente la fase como parte del criterio de reconstrucción, en lugar de operar únicamente sobre intensidades.

En cuanto al autoencoder, existen varias oportunidades de mejora. El uso de arquitecturas totalmente convolucionales permitiría eliminar el problema del mosaico de parches, recuperando continuidad en toda la imagen. Modelos tipo U-Net, variational autoencoders o redes generativas condicionadas podrían capturar estructuras interferométricas globales sin depender del entrenamiento local por bloques. Adicionalmente, se podría implementar un esquema de entrenamiento en dos etapas: primero con amplitud (como se realizó) y posteriormente con un

modelo específico para reconstruir la fase. Otra alternativa es emplear técnicas recientes como autoencoders perceptuales o pérdidas basadas en similitud estructural adaptada al dominio interferométrico.

Finalmente, el uso de datasets más amplios, incluso generados sintéticamente mediante simuladores de interferencia, podría aportar variabilidad suficiente para que los modelos aprendan estructuras complejas que no están presentes en un conjunto reducido de cien imágenes por desplazamiento.

Conclusiones

El trabajo desarrollado permitió implementar y evaluar un sistema completo para la reconstrucción de imágenes interferométricas mediante una combinación de técnicas de sparse coding y modelos generativos. A diferencia de los enfoques predominantes en el estado del arte actual que utilizan arquitecturas convolucionales directas como la U-Net para super resolución [102] o la Y-Net para reconstrucción mutiescala [23, 103], esta investigación exploro la capacidad de representación de diccionarios optimizados. La búsqueda evolutiva de hiperparámetros proporcionó configuraciones adecuadas para el entrenamiento de diccionarios, permitiendo realizar reconstrucciones visualmente coherentes y eliminando artefactos relevantes como el reflejo inicial. Los diccionarios entrenados con veinte y cien ejemplos demostraron estabilidad, y las reconstrucciones obtenidas alcanzaron valores promedio de PSNR alrededor de 25.5 dB y correlaciones superiores a 0.87, lo que indica que el MP fue capaz de reproducir adecuadamente la estructura general de las imágenes.

Sin embargo, el análisis detallado evidenció que, a diferencia de modelos especializados en fase como la MPRNet [21], que logra una restauración progresiva en múltiples etapas, esta aproximación no preserva la información de fase con suficiente precisión. Los bajos valores obtenidos en la comparación entre las fases reconstruidas y originales muestran que el método es adecuado para recuperar intensidad, pero no para obtener parámetros interferométricos avanzados que dependen de variaciones locales finas. Esto delimita el alcance del MP como herramienta para tareas de reconstrucción óptica más profundas frente a redes que utilizan funciones de perdida específicas para fase como la implementada por Jeon et al. [104] para la reducción de ruido speckle.

El autoencoder, aunque presentó avances cuando se entrenó con imágenes reconstruidas, no logró generar imágenes con resolución, continuidad ni fidelidad suficientes para permitir el cálculo de fase, desafío que autores como Luo et al. [102] mitigan mediante el uso de redes de super resolución de pixeles. La fuerte dependencia del muestreo por parches impuso una estructura cuadriculada en todas las predicciones, lo que compromete la utilidad del modelo en este contexto. Aun así, los resultados obtenidos señalan que la información interferométrica es aprendible parcialmente por redes neuronales, abriendo la posibilidad de desarrollar arquitecturas más adecuadas en estudios futuros.

En conjunto, el proyecto demuestra que la combinación de Matching Pursuit y aprendizaje profundo constituye una herramienta prometedora para la recuperación de información óptica, pero requiere mejoras importantes para alcanzar precisión interferométrica. Las conclusiones establecen un punto de partida sólido para futuras investigaciones orientadas a la recuperación directa de fase mediante redes neuronales, optimizaciones de diccionarios y modelos híbridos que integren las ventajas de ambos enfoques.

Publicaciones derivadas de este trabajo de tesis

- Hernández Flores, M. del R., Sánchez Rinza, B. E., & Rossainz López, M. (2025). Generación de imágenes holográficas sintéticas con desplazamiento de fase mediante autoencoders y preprocesamiento de ruido con Matching Pursuit. En Proceedings del 47 Congreso Internacional de Ingeniería en Electrónica (en prensa).

Bibliografía

[1] Beléndez Vázquez, A. Holografía: Generalidades. Fundamentos de Ótica para Ingeniería Infromática. Servicio de Publicaciones de la Universidad de Alicante (1996) ISBN: 84-7908-278-X. https://rua.ua.es/dspace/bitstream/10045/11865/1/Holografía_Generalidades.pdf

[2] Einstein, A., Podolsky, B., & Rosen, N. (1935). Can quantum-mechanical description of physical reality be considered complete? Physical Review, 47(10), 777-780. https://doi.org/10.1103/PhysRev.47.777

[3] Kwiat, P. G., Mattle, K., Weinfurter, H., Zeilinger, A., Sergienko, A. V., & Shih, Y. (1995). New high-intensity source of polarization-entangled photon pairs. Physical Review Letters, 75(24), 4337-4341. https://doi.org/10.1103/PhysRevLett.75.4337

[4] Lemos, G. B., Borish, V., Cole, G. D., Ramelow, S., Lapkiewicz, R., & Zeilinger, A. (2014). Quantum imaging with undetected photons. Nature, 512(7515), 409-412. https://doi.org/10.1038/nature13586

[5] Aspden, R. S., Morris, P. A., Bell, J. E., Boyd, R. W., & Padgett, M. J. (2015). Heralded phase-contrast imaging using an undetected photon. Nature Photonics, 9(9), 537-542. https://doi.org/10.1038/nphoton.2015.129

[6] Brida, G., Genovese, M., & Berchera, I. R. (2010). Experimental realization of sub-shot-noise quantum imaging. Nature Photonics, 4(4), 227-230. https://doi.org/10.1038/nphoton.2010.29

[7] Schnars, U., & Jüptner, W. (2002). Digital recording and numerical reconstruction of holograms. Measurement Science and Technology, 13(9), R85-R101. https://doi.org/10.1088/0957-0233/13/9/201

[8] Kim, M. K. (2010). Principles and applications of digital holography. John Wiley & Sons.

[9] Picart, P., Leval, J., Mounier, D., & Gougeon, S. (2003). Digital holography using a CCD camera and numerical reconstruction. Optics and Lasers in Engineering, 38(1-2), 141-150. https://doi.org/10.1016/S0143-8166(02)00065-4

[10] Yamaguchi, I., & Zhang, T. (1997). Phase-shifting digital holography. Optics Letters, 22(16), 1268-1270. https://doi.org/10.1364/OL.22.001268

[11] Xu, W., Jericho, M. H., Kreuzer, H. J., & Gauthier, R. C. (2001). Digital in-line holography for biological applications. Proceedings of the National Academy of Sciences, 98(20), 11301-11305. https://doi.org/10.1073/pnas.191361398

[12] Goodman, J. W. (2005). Introduction to Fourier optics (3rd ed.). Roberts & Company Publishers.

[13] Marquet, P., Rappaz, B., Magistretti, P. J., Cuche, E., Emery, Y., Colomb, T., & Depeursinge, C. (2005). Digital holographic microscopy: a noninvasive contrast imaging technique allowing quantitative visualization of living cells with subwavelength axial accuracy. Optics Letters, 30(5), 468-470. https://doi.org/10.1364/OL.30.000468

[14] Vest, C. M. (1979). Holographic interferometry. Wiley.

[15] Poon, T. C. (2006). Digital holography and three-dimensional display: Principles and applications. Springer.

[16] Kemper, B., & Von Bally, G. (2008). Digital holographic microscopy for live cell applications and technical inspection. Applied Optics, 47(4), A52-A61. https://doi.org/10.1364/AO.47.000A52

[17] Zhang, Q. N. et al. (2020). Deep phase shifter for quantitative phase imaging. Pre - print at https://doi.org/10.48550/arXiv.2003.03027.

[18] Zhang, Q. N. et al. (2021). Phase-shifting interferometry from single frame in-line interferogram using deep learning phase-shifting technology. Opt. Commun. 498, 127226.

[19] Wang, K. Q. et al. (2019). Y-Net: a one-to-two deep learning framework for digital holographic reconstruction. Opt. Lett. 44,4765–4768.

[20] Yan, K. T. etal. (2022). Virtual temporal phase-shifting phase extraction using generative adversarial networks. Appl. Opt. 61,2525–2535.

[21] Zhao, Y., Hu, K. & Liu, F. W. (2023). One-shot phase retrieval method for interferometry using a multi-stage phase-shifting network. IEEE Photonics Technol. Lett. 35,577–580.

[22] Zamir, S. W. et al. Multi-Stage Progressive Image Restoration. in Proceedings of 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition 14821–14831 (IEEE, 2021).

[23] Huang, T. et al. (2023). Single-shot Fresnel incoherent correlation holography via deep learning based phase-shifting technology. Opt. Express 31, 12349–12356.

[24] Wu, B. et al. (2023). RSAGAN: Rapid self-attention generative adversarial nets for single-shot phase-shifting interferometry. Opt. Lasers Eng. 168, 107672.

[25] Luo, H. et al. (2022). Diffraction-Net: a robust single-shot holography for multi-distance lensless imaging. Opt. Express 30,41724–41740.

[26] Li, J. S. et al. (2020). Quantitative phase imaging in dual-wavelength interferometry using a single wavelength illumination and deep learning. Opt. Express 28, 28140–28153.

[27] Li, J. S. et al. (2021). Hybrid-net: a two-to-one deep learning framework for three wavelength phase-shifting interferometry. Opt. Express 29, 34656–34670.

[28] Xu, X. Q. et al. (2021). Dual-wavelength interferogram decoupling method for three frame generalized dual-wavelength phase-shifting interferometry based on deep learning. J. Opt. Soc. Am. A 38,321–327.

[29] Born, M. &Wolf, E. Principles of Optics: Electromagnetic Theory Of Propagation, Interference And Diffraction Of Light. 6th edn (Pergamon Press, 1980).

[30] Shechtman, Y. et al. Phase retrieval with application to optical imaging: a contemporary overview. IEEE Signal Process. Mag. 32,87–109 (2015).

[31] Park, Y., Depeursinge, C. & Popescu, G. Quantitative phase imaging in biomedicine. Nat. Photonics 12, 578–589 (2018).

[32] Miao, J. W. et al. Extending the methodology of X-ray crystallography to allow imaging of micrometre-sized non-crystalline specimens. Nature 400, 342–344 (1999).

[33] Tyson, R. K. & Frazier, B. W. Principles of Adaptive Optics.5th edn (CRCPress, 2022).

[34] Colomb, T. & Kühn, J. Digital holographic microscopy. in Optical Measurement of Surface Topography (ed. Leach, R.) 209–235 (Springer, 2011).

[35] Goodman, J. W. Introduction to Fourier Optics. 4th edn (W.H. Freeman, 2017).

[36] Gabor, D. A new microscopic principle. Nature 161,777–778 (1948).

[37] Gerchberg, R.W. & Saxton,W.O. A practical algorithm for the determination of phase from image and diffraction plane picture. Optik 35,237–246 (1972).

[38] Fienup, J. R. Phase retrieval algorithms: a comparison. Appl. Opt. 21, 2758–2769 (1982).

[39] Fienup, J. R. Reconstruction of an object from the modulus of its Fourier transform. Opt. Lett. 3,27–29 (1978).

[40] Allen, L. J. & Oxley, M. P. Phase retrieval from series of images obtained by defocus variation. Opt. Commun. 199,65–75 (2001).

[41] Pedrini, G., Osten, W. & Zhang, Y. Wave-front reconstruction from a sequence of interferograms recorded at different planes. Opt. Lett. 30,833–835 (2005).

[42] Greenbaum, A. & Ozcan, A. Maskless imaging of dense samples using pixel super-resolution based multi-height lensfree on-chip microscopy. Opt. Express 20,3129–3143 (2012).

[43] Hoppe, W. & Strube, G. Beugung in inhomogenen Primärstrahlenwellenfeld. II. Lichtoptische Analogieversuche zur Phasenmessung von Gitterinterferenzen. Acta Crystallogr. Sect. A 25,502–507 (1969).

[44] Faulkner, H. M. L. & Rodenburg, J. M. Movable aperture lensless transmission microscopy: a novel phase retrieval algorithm. Phys. Rev. Lett. 93, 023903 (2004).

[45] Rodenburg, J.M. & Faulkner, H. M. L. A phase retrieval algorithm for shifting illumination. Appl. Phys. Lett. 85,4795–4797 (2004).

[46] Zheng, G. A., Horstmeyer, R. & Yang, C. H. Wide-field, high-resolution Fourier ptychographic microscopy. Nat. Photonics 7,739–745 (2013).

[47] Zheng, G. A. et al. Concept, implementations and applications of Fourier ptychography. Nat. Rev. Phys. 3,207–223 (2021).

[48] Zhang, K. Y. J. & Main, P. Histogram matching as a new density modification technique for phase refinement and extension of protein molecules. Acta Crystallogr. Sect. A: Found. Crystallogr. 46,41–46 (1990).

[49] Elser, V. Solution of the crystallographic phase problem by iterated projections. Acta Crystallogr. Sect. A Found. Crystallogr. 59,201–209 (2003).

[50] Latychevskaia, T. & Fink, H.-W. Solution to the twin image problem in holography. Phys.Rev. Lett.98, 233901 (2007).

[51] Moravec, M. L., Romberg, J. K. & Baraniuk, R. G. Compressive phase retrieval. in Proceedings of SPIE 6701, Wavelets XII. 670120 (SPIE, 2007).

[52] Kostenko, A. et al. Phase retrieval in in-line x-ray phase contrast imaging based on total variation minimization. Opt. Express 21, 710–723 (2013).

[53] Gao, Y. H. &Cao, L. C. Iterative projection meets sparsity regularization: towards practical single-shot quantitative phase imaging with in-line holography. Light Adv. Manuf. 4,37–53 (2023).

[54] Rivenson, Y. et al. Sparsity-based multi-height phase recovery in holographic microscopy. Sci. Rep. 6, 37862 (2016).

[55] Candès, E. J., Li, X. D. & Soltanolkotabi, M. Phase retrieval via Wirtinger flow: theory and algorithms. IEEE Trans. Inf. Theory 61,1985–2007 (2015).

[56] Wang, G., Giannakis, G. B. & Eldar, Y. C. Solving systems of random quadratic equations via truncated amplitude flow. IEEE Trans. Inf. Theory 64, 773–794 (2018).

[57] Candès, E. J., Strohmer, T. & Voroninski, V. PhaseLift: exact and stable signal recovery from magnitude measurements via convex programming. Com mun. Pure Appl. Math. 66,1241–1274 (2013).

[58] Kreis, T. (2005). Handbook of holographic interferometry: Optical and digital methods. Wiley-VCH.

[59] Schnars, U., & Jüptner, W. (2005). Digital holography: Digital hologram recording, numerical reconstruction, and related techniques. Springer.

[60] Kim, M. K. (2010). Digital holographic microscopy: Principles, techniques, and applications. Springer.

[61] Sinha, A., Lee, J., Li, S., & Barbastathis, G. (2017). Lensless computational imaging through deep learning. Optica, 4(9), 1117-1125. https://doi.org/10.1364/OPTICA.4.001117

[62] Jain, A. K. (1989). Fundamentals of digital image processing. Prentice-Hall.

[63] Gonzalez, R. C., & Woods, R. E. (2018). Digital image processing (4th ed.). Pearson.

[64] Mallat, S. (1999). A wavelet tour of signal processing. Academic Press.

[65] Zhang, K., Zuo, W., Chen, Y., Meng, D., & Zhang, L. (2017). Beyond a Gaussian denoiser: Residual learning of deep CNN for image denoising. IEEE Transactions on Image Processing, 26(7), 3142-3155.

[66] Aspden, R. S., Tasca, D. S., Boyd, R. W., & Padgett, M. J. (2015). EPR-based ghost imaging using a single-photon-sensitive camera. New Journal of Physics, 17(7), 073032.

[67] Mallat, S., & Zhang, Z. (1993). Matching Pursuits with time-frequency dictionaries. IEEE Transactions on Signal Processing, 41(12), 3397-3415.

[68] Tropp, J. A. (2004). Greed is good: Algorithmic results for sparse approximation. IEEE Transactions on Information Theory, 50(10), 2231-2242.

[69] Chen, S. S., Donoho, D. L., & Saunders, M. A. (2001). Atomic decomposition by basis pursuit. SIAM Review, 43(1), 129-159.

[70] Tropp, J. A., & Gilbert, A. C. (2007). Signal recovery from random measurements via orthogonal Matching Pursuit. IEEE Transactions on Information Theory, 53(12), 4655-4666.

[71] Donoho, D. L. (2006). Compressed sensing. IEEE Transactions on Information Theory, 52(4), 1289-1306.

[72] Mallat, S. (2009). A Wavelet Tour of Signal Processing: The Sparse Way. Academic Press.

[73] Daubechies, I. (1992). Ten Lectures on Wavelets. SIAM.

[74] Feichtinger, H. G., & Strohmer, T. (1998). Gabor Analysis and Algorithms: Theory and Applications. Birkhäuser.

[75] Aharon, M., Elad, M., & Bruckstein, A. (2006). The K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation. IEEE Transactions on Signal Processing, 54(11), 4311-4322.

[76] Elad, M. (2010). Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing. Springer.

[77] Tosic, I., & Frossard, P. (2011). Dictionary Learning. IEEE Signal Processing Magazine, 28(2), 27-38.

[78] Elad, M., & Aharon, M. (2006). Image denoising via sparse and redundant representations over learned dictionaries. IEEE Transactions on Image Processing, 15(12), 3736-3745.

[79] Wright, J., Yang, A. Y., Ganesh, A., Sastry, S. S., & Ma, Y. (2010). Robust face recognition via sparse representation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 31(2), 210-227.

[80] Szu, H. H., Dong, J., & Burge, M. (2007). Holographic neural technology for image recognition. Optical Engineering, 46(5), 057001.

[81] Rubinstein, R., Zibulevsky, M., & Elad, M. (2010). Efficient implementation of the K-SVD algorithm using batch orthogonal Matching Pursuit. IEEE Transactions on Signal Processing, 57(12), 5695-5702.

[82] Donoho, D. L., Tsaig, Y., Drori, I., & Starck, J. L. (2012). Sparse solution of underdetermined systems of linear equations by stagewise orthogonal Matching Pursuit. IEEE Transactions on Information Theory, 58(2), 1094-1121.

[83] Donoho, D. L., Maleki, A., & Montanari, A. (2009). Message-passing algorithms for compressed sensing. Proceedings of the National Academy of Sciences, 106(45), 18914-18919.

[84] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press.

[85] Aggarwal, C. C. (2018). Neural networks and deep learning. Springer.

[86] Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. Psychological Review, 65(6), 386-408.

[87] Rumelhart, D., Hinton, G. & Williams, R. Learning representations by back-propagating errors. Nature 323, 533–536 (1986). https://doi.org/10.1038/323533a0.

[88] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep Learning. Nature, 521(7553), 436-444. https://doi.org/10.1038/nature14539

[89] Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. Science, 313(5786), 504-507.

[90] Kingma, D. P., & Welling, M. (2013). Auto-encoding variational Bayes. arXiv preprint arXiv:1312.6114.

[91] Vincent, P., Larochelle, H., Bengio, Y., & Manzagol, P. A. (2010). Extracting and composing robust features with denoising autoencoders. Neural computation, 22(5), 1236-1262.

[92] Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks. In Proceedings of the 14th International Conference on Artificial Intelligence and Statistics (pp. 315-323).

[93] Wang, Z., & Bovik, A. C. (2006). Image Quality Assessment: From Error Visibility to Structural Similarity. IEEE Transactions on Image Processing, 13(4), 600-612.

[94] Molina, J., Garcia, F., & Romero, M. (2015). A Study of Image Quality Metrics in Various Contexts. Wiley.

[95] Zhu, L., Zhang, L., & Wu, L. (2019). Evaluation of Image Quality in Digital Holography: A Quantitative Approach. Journal of Optical Society of America A, 36(9), 1340-1349.

[96] Gonzalez, R. C., & Woods, R. E. (2008). Digital Image Processing (3rd ed.). Pearson Prentice Hall.

[97] Wang, Z., & Bovik, A. C. (2009). Mean squared error: Love it or leave it? IEEE Signal Processing Magazine, 26(1), 98-117. https://doi.org/10.1109/MSP.2008.930649

[98] Hassan, M. I., & Sher, S. (2011). Evaluation of Image Quality Metrics for Digital Images. Springer.

[99] Hore, A., & Ziou, D. (2010). Image quality metrics: PSNR vs. SSIM. 2010 20th International Conference on Pattern Recognition, 2366–2369. https://doi.org/10.1109/ICPR.2010.579

[100] Valin-Rivera, J., Monteiro, J., Pires-Vaz, M. A., Lopes, H., Teixeira-Coelho, R., Martinez-Aneiros, F., Valin-Fernández, M., & Gonçalves, E. (2017). Holografía digital aplicada para la evaluación de pastilla de metal duro. Ingeniería mecánica, 20(1), 39–44. http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S1815-59442017000100006

[101] Marín-Velásquez, T. D. (2024). Tendencia corrosiva por CO2 del gas natural basada en su composición mediante Redes Neuronales Artificiales. FIGEMPA: Investigación y Desarrollo, 18(2), 1–13. https://doi.org/10.29166/revfig.v18i2.5989

[102] Luo, Z. X. et al. Pixel super-resolution for lens-free holographic microscopy using deep learning neural networks. Opt. Express 27, 13581–13595 (2019)

[103] Zhang, J. Z. et al. Fourier ptychographic microscopy reconstruction with multiscale deep residual network. Opt. Express 27,8612–8625 (2019).

[104] Jeon, W. et al. Speckle noise reduction for digital holographic images using multi-scale convolutional neural networks. Opt. Lett. 43,4240–4243 (2018).