Universidad de Oviedo

ESCUELA POLITÉCNICA DE INGENIERÍA DE GIJÓN

GRADO EN INGENIERÍA INFORMÁTICA EN TECNOLOGÍAS DE LA INFORMACIÓN

ÁREA DE INGENIERÍA TELEMÁTICA

Utilización de un sistema basado en Deep Learning para reconstrucción tomográfica con la técnica WOMBAT

ÁLVAREZ ÁLVAREZ, ALBERTO
TUTORES:
Carlos González Gutiérrez
Alejandro Buendía Roca

FECHA: 26 de julio de 2024

Índice

Índice de figuras

Índice de tablas

1. Introducción y estructura del documento

1.1. INTRODUCCIÓN

La reconstrucción tomográfica es una técnica avanzada que ha encontrado aplicaciones significativas en diversos campos científicos, incluyendo la astronomía. Una de las técnicas más prometedoras en este ámbito es WOMBAT (*Wavefront Obtained from Measurements from Beam-profiles through Atmospheric Turbulences*) la cual, combinada con sistemas basados en *deep learning*, ofrece mejoras sustanciales en la precisión y eficiencia de los procesos de imagen en telescopios de gran tamaño.

La atmósfera terrestre, con su composición variable y turbulencias, representa un desafío considerable para la observación astronómica. Estas turbulencias distorsionan el frente de onda de la luz proveniente de objetos celestes, resultando en imágenes borrosas y con pérdida de detalles. Para mitigar estos efectos, se han desarrollado técnicas de óptica adaptativa que utilizan modelos matemáticos de la atmósfera y herramientas ópticas avanzadas para corregir las distorsiones en tiempo real.

En este punto se sustenta la realización del presente trabajo, el cual se trata de la utilización de un sistema basado en *deep learning* para la reconstrucción tomográfica con la técnica WOMBAT. Esta técnica destaca por su capacidad para mapear y analizar campos ópticos de manera eficiente, utilizando análisis binario que facilita la identificación y corrección de aberraciones en el frente de onda. Cuando esta técnica se integra con algoritmos de *deep learning*, se potencia su efectividad. Los algoritmos de aprendizaje profundo, entrenados con grandes volúmenes de datos de observación, son capaces de aprender patrones complejos de distorsión y predecir las correcciones necesarias con alta precisión.

La combinación de WOMBAT y *deep learning* en la reconstrucción tomográfica ofrece numerosas ventajas clave, las cuales se tratarán de aprovechar. En primer lugar, mejora significativamente la precisión de las imágenes. Los modelos de este tipo tienen la

capacidad de adaptarse continuamente a las condiciones atmosféricas cambiantes, proporcionando correcciones precisas que resultan en imágenes más nítidas y detalladas. Además, la eficiencia en el procesamiento de datos se ve considerablemente aumentada.

El procesamiento automatizado y en tiempo real de grandes volúmenes de datos de imagen permite un análisis más rápido y eficiente, lo que facilita la toma de decisiones en tiempo real durante las observaciones astronómicas.

Por último, esta combinación reduce los costos operativos. Al mejorar la calidad de las imágenes obtenidas desde telescopios terrestres, se minimiza la necesidad de costosas misiones espaciales para obtener observaciones sin distorsiones atmosféricas.

A pesar de sus beneficios, la integración de WOMBAT y el aprendizaje profundo en la reconstrucción tomográfica presenta desafíos técnicos y computacionales. El entrenamiento de modelos de aprendizaje automático avanzado requiere acceso a grandes conjuntos de datos y capacidades de procesamiento significativas. Además, la implementación práctica de estas técnicas en sistemas de telescopios existentes puede requerir adaptaciones y mejoras en la infraestructura tecnológica. Por lo tanto, estos aspectos representan los principales retos que se abordarán a lo largo de este estudio.

1.2. ESTRUCTURA DEL DOCUMENTO

Este documento está estructurado en varios capítulos que abordan de manera detallada los distintos aspectos del trabajo realizado. A continuación, se describe la organización y el contenido de cada uno de los capítulos.

El primer capítulo, "Introducción y estructura del documento", proporciona una visión general del trabajo y explica cómo está organizado el documento. En la sección de introducción, se presenta el contexto y la motivación del trabajo, así como los objetivos generales que se buscan alcanzar.

El segundo capítulo, "Objetivos del trabajo", detalla los objetivos específicos del proyecto. Aquí se discuten las metas a corto y largo plazo, así como los resultados esperados al finalizar el estudio.

En el tercer capítulo, "Fundamentos teóricos", se presentan los conceptos y teorías fundamentales que sustentan el trabajo. Este capítulo se divide en dos partes principales. La primera parte se centra en la óptica adaptativa, abordando los principios de esta disciplina, con un enfoque en los efectos de la atmósfera y los sistemas utilizados para mitigar estos efectos. La segunda parte introduce el concepto de redes neuronales y su relevancia para la óptica adaptativa. Se proporciona una visión general de las redes neuronales, incluyendo su estructura y funcionamiento básico, se discute el uso de tarjetas gráficas para la aceleración del entrenamiento y la implementación de redes neuronales, y se exploran otras aplicaciones de estas técnicas en distintos campos.

El cuarto capítulo, "Herramientas y tecnologías", describe las herramientas y tecnologías utilizadas en el desarrollo del trabajo. Se enumeran y describen los componentes de hardware empleados en el proyecto, el lenguaje de programación seleccionado y sus ventajas para este tipo de trabajo, así como las librerías y el entorno de trabajo utilizados para implementar las soluciones propuestas.

El quinto capítulo, "Obtención de los datos", explica el proceso de recolección de datos necesarios para el desarrollo y validación del modelo. Se describe el sistema WOMBAT y su papel en la obtención de datos atmosféricos, así como la simulación de

datos atmosféricos para complementar los datos reales y asegurar una cobertura amplia de escenarios.

El sexto capítulo, "Metodología de trabajo", expone la metodología seguida para el desarrollo del trabajo. Se describe el proceso de selección del modelo base sobre el cual se realizaron las modificaciones y experimentos, y se presentan los distintos conjuntos de datos utilizados en el entrenamiento y validación del modelo.

En el séptimo capítulo, "Experimentos y resultados", se detallan los experimentos realizados y los resultados obtenidos. Se presentan los resultados iniciales obtenidos utilizando el modelo base y se exploran distintas modificaciones al modelo base y sus impactos en los resultados. Estas variaciones incluyen el uso de diferentes funciones de activación, técnicas de normalización, ajustes en el tamaño de los filtros de las capas convolucionales, cambios en la tasa de *dropout*, y la adición o eliminación de capas convolucionales y totalmente conectadas. Finalmente, se recopilan y discuten los resultados obtenidos de los diferentes experimentos, comparando el rendimiento de las distintas configuraciones.

Por último, el octavo capítulo, "Conclusiones y trabajos futuros", presenta las conclusiones derivadas del trabajo realizado y propone posibles líneas de investigación futura.

2. Objetivos del trabajo

El objetivo definido para la realización del presente proyecto se basa en la utilización de redes neuronales convolucionales (CNN) para la corrección de turbulencias atmosféricas en la observación óptica, empleando el sistema WOMBAT para la obtención de datos. De este modo, el proyecto se puede dividir en diferentes fases, cuyo desarrollo permite el avance hacia la siguiente fase.

El comienzo del proyecto se centra en la introducción y estudio del funcionamiento de las redes neuronales convolucionales y sus aplicaciones en la corrección de turbulencias ópticas, siendo el objetivo conocer las ventajas que aporta dicho modelo de aprendizaje frente a los métodos tradicionales y conseguir las aptitudes para poder utilizarlo.

Tras esta primera fase de introducción general a las nuevas tecnologías, se entra de lleno en la aplicación de estas en un problema concreto: la corrección de turbulencias atmosféricas en la observación óptica. Es decir, se fija el objetivo final del proyecto que consiste en realizar un sistema de corrección de turbulencias partiendo de un conjunto de datos complejos y reales proporcionados por WOMBAT.

Los datos utilizados en este proyecto se caracterizan por simular las condiciones atmosféricas turbulentas que afectan la calidad de las imágenes ópticas. La obtención de estos datos se realiza mediante WOMBAT, complementado con simulaciones que aseguran una cobertura amplia de escenarios. La complejidad de estos datos radica en la variabilidad de las condiciones atmosféricas, como la iluminación y las características de las turbulencias, lo que añade un nivel de desafío significativo para cualquier sistema de corrección.

Siendo así, se pretende desarrollar un sistema de corrección de turbulencias haciendo uso de un modelo de CNN razonado y estudiado, acorde a dicha tarea, y cuyo rendimiento se asemeje a la corrección que realizaría un sistema óptico ideal. Por lo tanto, se deberá de realizar un estudio del estado del arte enfocado en la corrección de turbulencias basado en CNNs.

Finalmente, tras dar con el enfoque de modelo apropiado, se llevará este a la práctica, realizando un plan de experimentación en busca de posibles mejoras en el modelo, encontrando aquel que proporcione el mayor rendimiento al sistema.

En resumen, los objetivos fijados son:

- Estudio tanto teórico como práctico de las CNNs, adquiriendo los conocimientos necesarios para realizar posteriormente el desarrollo del sistema.
- Realización de una implementación completa de un sistema de corrección de turbulencias basado en CNNs, pudiendo obtener una mejora significativa en la calidad de las imágenes y resultados.
- Plan de experimentación de los modelos encontrados, seleccionando aquel que proporcione el rendimiento más alto en base a las métricas comúnmente utilizadas en este tipo de problemas.

3. Fundamentos teóricos

La reconstrucción tomográfica utilizando la técnica WOMBAT y sistemas basados en *deep learning* puede mejorar la precisión y eficiencia de los procesos de imagen en telescopios de gran tamaño.

3.1. ÓPTICA ADAPTATIVA

La atmósfera ha supuesto un obstáculo a la hora de observar con claridad los eventos ocurridos en el firmamento. Como posible solución para estos problemas surge la llamada óptica adaptativa, la cual serviría para eliminar, en la medida de lo posible, los diferentes efectos causados por las turbulencias atmosféricas que se podrían encontrar al realizar las observaciones.

Para conseguir solucionar los problemas derivados de las turbulencias ha sido necesario realizar una caracterización matemática de la atmósfera, gracias a lo que se consiguió crear varios modelos que facilitarían su comprensión. Para corregir las diferentes imágenes los telescopios modernos hacen uso de una amplia gama de principios y fenómenos de la óptica geométrica utilizando diferentes herramientas y elementos para obtener la mejor vista del espacio desde la tierra.

3.1.1. La atmósfera

La atmósfera está compuesta por un 78% de nitrógeno, un 21% de oxígeno y el resto es una mezcla de diferentes gases como argón, dióxido de carbono, neón, helio, etc. El polvo y el vapor de agua también forman parte de la atmósfera. Esta composición ha permitido la creación, desarrollo y evolución de la vida en nuestro planeta y protege de diferentes radiaciones y regula la temperatura.

Pese a todas las ventajas que otorga, genera diferentes problemas a las observaciones astronómicas, las denominadas turbulencias atmosféricas. Cambios en aspectos como la presión, la temperatura y la mezcla de gases pueden resultar en variaciones de los diferentes índices de refracción y absorción, que afectan directamente a cómo la luz llega a los diferentes dispositivos. Estos índices alteran la trayectoria de los haces de luz provenientes de un mismo objeto, lo que finalmente produce una distorsión en el frente de onda. Esta distorsión puede causar un efecto de desenfoque, pérdida de detalles o borrosidad en las imágenes captadas.

Como ayuda para contrarrestar estas turbulencias, y siendo clave para el total de este trabajo, se introducen entonces los polinomios de Zernike, representación matemática con similitudes a las series de Taylor o de Fourier, pero sobre una superficie circular. Estos polinomios son especialmente útiles en óptica para describir las aberraciones de frente de onda en sistemas ópticos con pupilas circulares. Al ser una serie de funciones ortogonales y continúas definidas sobre un círculo de unidad, cada coeficiente en la expansión de Zernike representa de forma independiente una característica particular de la aberración del frente de onda. Esta base matemática en coordenadas polares permite que cada polinomio se exprese como una combinación de funciones radiales y angulares independientes entre sí, facilitando así la descomposición y la reconstrucción precisa de los frentes de onda. Además, los polinomios de Zernike son una herramienta valiosa para calcular las diferentes alteraciones que ocurren cuando la luz atraviesa la atmósfera, permitiendo correcciones ópticas eficaces y específicas en tecnologías como la óptica adaptativa. [1]

3.1.1.1. Frente de onda

Un frente de onda se describe comúnmente como el conjunto de puntos que están en la misma fase de una onda o como una superficie imaginaria que representa los puntos de una onda que vibran simultáneamente. A medida que la onda se propaga, el frente de onda se desplaza a la velocidad de la onda. [2]

Esta definición permite afirmar que, cuando la luz de un objeto suficientemente lejano se acerca a la tierra, su frente de onda es casi completamente plano, ya que en el espacio exterior no hay elementos que generen interferencias en él. Esta afirmación se explica porque la esfera generada por un objeto puntual es tan grande que el fragmento que llega se puede considerar plano.

En el punto en que dicha luz proveniente del objeto entra en la atmósfera terrestre y la atraviesa, los diferentes fotones van llegando de manera continua pero desfasada, es decir, con un frente de onda no plano. Debido a la naturaleza de la atmósfera, esto genera una imagen borrosa en los instrumentos, de la cual tal vez no se pueda extraer ningún uso útil.

Para representar un frente de onda, una de las técnicas más simples es emplear una imagen discreta $W\ (x,\ y)$ de dos dimensiones, donde cada punto representa el desfase respecto al frente de onda plano. Esta representación facilita el cálculo del error cuadrático medio del frente de onda, también conocido como WFE.

$$WFE = \sqrt{\frac{\sum_{i=0,j=0}^{x,y}(W(i,j)-\overline{W})^2}{x \cdot y}} \quad (1)$$

En esta ecuación, $\overline{W}$ es el promedio de todos los valores y $(x,\ y)$ es el número total de píxeles que la imagen tiene en cada una de sus dimensiones.

El resultado se expresa generalmente en las unidades de la longitud de onda de la luz observada (del orden de los nanómetros) y permite determinar cómo de alejado se encuentra el frente de onda de la representación plana, por lo que es posible obtener cómo la atmósfera ha modificado el frente de onda.

Otro tipo de representación son los polinomios de Zernike, los cuales son una representación mediante polinomios ortonormales a lo largo de un círculo unidad. Esto es debido a que las lentes de los telescopios mediante las que se reciben los frentes de onda son pupilas circulares.

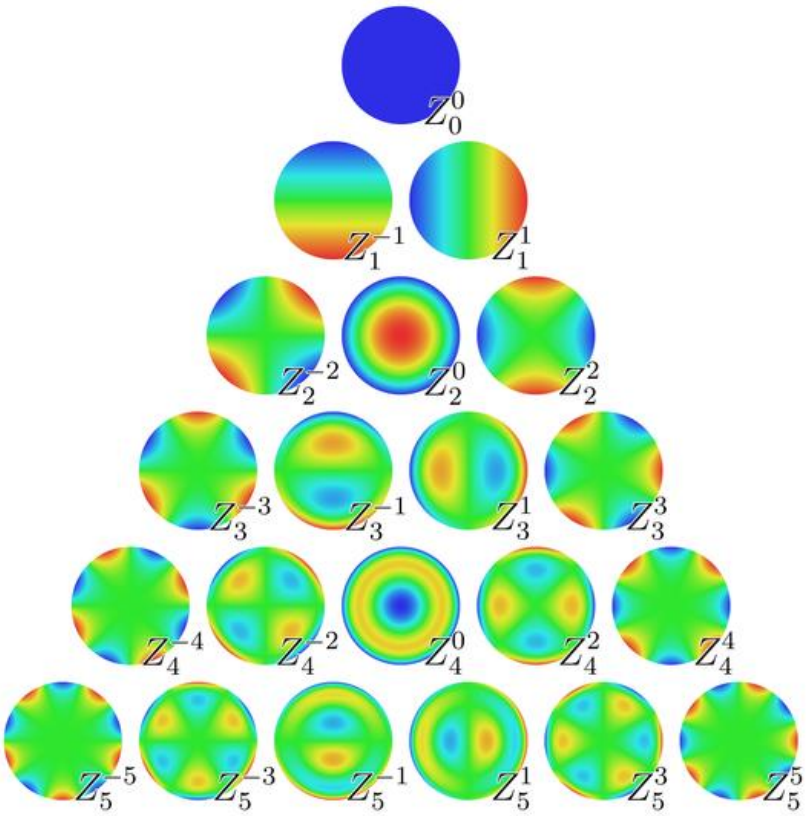

Figura 3.1- Primeros 21 polinomios de Zernike, ordenados verticalmente por grado radial y horizontalmente por grado azimutal.

En la Figura 3.1 se muestra los primeros 21 polinomios de Zernike, ordenados verticalmente por grado radial y horizontalmente por grado azimutal. Cada círculo en la figura representa un polinomio diferente, donde los colores indican las variaciones de fase: las áreas rojas y verdes representan las diferencias de fase positiva y negativa, respectivamente.

Mediante estos polinomios, es posible reconstruir el frente de onda que ha pasado por la turbulencia atmosférica utilizando las llamadas series de Zernike. En estas series, el frente de onda recuperado $W\ (r,\ \theta)$, ahora representado en coordenadas polares, se puede definir de la siguiente manera:

$$W(r,\theta) = \sum_{n,m} A_n^m Z_n^m\ (r,\theta) \qquad (2)$$

En este contexto, $Z_n^m(r,\ \theta)$ se refiere a los polinomios de Zernike, y A_n^m son los coeficientes de Zernike. De este modo, al contar con un conjunto de polinomios ya

establecidos, es posible reconstruir cualquier frente de onda simplemente determinando sus coeficientes.

3.1.1.2. Modelos atmosféricos

La caracterización matemática de la atmósfera ha permitido crear modelos, como el de turbulencia atmosférica, en el cual es común utilizar una representación de los distintos índices de refracción que existen sobre la vertical de un punto específico, a menudo descrito como el perfil C_n^2, la constante de estructura de la turbulencia atmosférica.

Aunque estos perfiles son representaciones continuas que contienen una infinidad de valores a lo largo de la vertical, una solución bastante común es modelizarlos utilizando un número finito de capas turbulentas extremadamente delgadas. Estas capas se ubican a diferentes alturas y poseen diferentes intensidades. Además, cada una de ellas tiene la capacidad de moverse en direcciones y velocidades distintas. Esta suposición, conocida como “la hipótesis del flujo congelado de Taylor”, permite simplificar la descripción de la turbulencia atmosférica al definir el conjunto total como la suma de las diferentes capas.

La hipótesis del flujo congelado de Taylor es un modelo fundamental en la descripción de la turbulencia atmosférica en estudios de óptica adaptativa. Este modelo supone que las capas turbulentas de la atmósfera, aunque en constante movimiento, pueden considerarse estáticas durante el corto tiempo que toma para que la luz las atraviese. Es decir, las distorsiones introducidas por estas capas se consideran "congeladas" en el tiempo, lo que permite simplificar matemáticamente el modelado de la propagación de la luz a través de la atmósfera. Este enfoque es crucial para diseñar sistemas de óptica adaptativa eficaces, ya que permite predecir y corregir las aberraciones ópticas en tiempo real, lo que es esencial para mejorar la calidad de las imágenes obtenidas desde telescopios terrestres [3]. Este enfoque será el que se adoptará en el desarrollo de este trabajo.

Existen varios parámetros que afectan a cada capa de la atmósfera, como su altura, velocidad y dirección de movimiento. Sin embargo, hay otro parámetro crucial para describir la turbulencia atmosférica, conocido como la longitud de coherencia de Fried o r_0. Se define como el diámetro de la apertura sobre el cual la varianza de la aberración de

fase es igual a un radián cuadrático medio. Este parámetro es fundamental en la óptica adaptativa, ya que establece la escala espacial sobre la cual se deben realizar correcciones para optimizar la calidad de las imágenes observadas bajo condiciones atmosféricas típicas.

Es importante destacar que el valor de r_0 no es constante, varía con la longitud de onda de la luz observada y aumenta con la altura del ángulo cenital. Además, este valor es inversamente proporcional a la variabilidad del índice de refracción atmosférico, indicando que, en condiciones de mayor turbulencia atmosférica, el r_0 será menor. Esto implica una necesidad de correcciones más frecuentes y precisas por parte del sistema de óptica adaptativa [3].

Este parámetro puede calcularse en función del perfil C_n^2 utilizando la ecuación:

$$r_0 = \left[0{,}423k^2 \cdot sec(\beta) \int_0^L C_n^2\,(z)dz\right]^{-3/5} \quad (3)$$

En esta ecuación, L representa la longitud del perfil o altura, dado que es la longitud del camino que sigue la luz, β es el ángulo del cenit, $C_n^2\,(z)$ representa cómo varía el perfil n con respecto a una altura específica z, y k es el número de onda. El parámetro r_0 sirve como un indicador de la resolución de observación que se puede lograr con un telescopio específico. Cuanto mayor sea este parámetro, mayor será la resolución. Por esta razón, la mayoría de los grandes observatorios se encuentran en lugares donde el parámetro r_0 es alto, es decir, donde la turbulencia es más débil y permite una mayor resolución, lo cual coincide con lugares con baja humedad y nubosidad y, sobre todo, con gran altura.

3.1.2. Sistemas de óptica adaptativa

La búsqueda de mecanismos que reduzcan los efectos introducidos por las turbulencias atmosféricas está presente en los diferentes telescopios desde mediados del siglo XX. Este fenómeno se observa tanto en telescopios refractores como en reflectores. Sin embargo, los telescopios reflectores son más comunes y forman la base de la mayoría de los grandes telescopios terrestres. En los telescopios reflectores, hay muchas variantes, pero todos comparten una característica común: la luz que entra al telescopio no pasa a

través de una lente. En cambio, el haz de luz se refleja a través de uno o más espejos curvados hasta que converge en un punto, creando la imagen que se observa.

A continuación, se definirán los diferentes elementos comunes entre los elementos empleados por los sistemas de óptica adaptativa, los cuales se usan para predecir, corregir y mejorar las imágenes obtenidas por el telescopio.

3.1.2.1. Sensor de frente de onda

El sensor de frente de onda es un componente fundamental en los sistemas de óptica adaptativa, encargado de medir las aberraciones del frente de onda de la luz. Utiliza estrellas guía, que pueden ser naturales o artificiales, para proporcionar una referencia de medición. Las estrellas guía naturales son aquellas que ya existen en el campo de visión del objeto a observar. Por otro lado, las estrellas guía artificiales se crean mediante láseres que se dirigen a puntos específicos en la atmósfera, generando un punto brillante que el sensor puede utilizar para medir las distorsiones del frente de onda causadas por las turbulencias atmosféricas. Esta medición precisa permite al sistema de óptica adaptativa corregir las aberraciones y mejorar la calidad de la imagen obtenida.

Esta luz se recogerá mediante un sensor fotográfico, generalmente de tipo CCD. La principal función de los sensores de este tipo es convertir la luz obtenida en señales eléctricas para así conseguir recrear imágenes digitales, para ser enviada a los reconstructores.

En el presente trabajo se utilizan sensores de frente de onda de tipo Shack-Hartmann (SH-WFS), los cuales tienen la capacidad de detectar las alteraciones en un frente de onda. Esto lo logra al dividir el frente de onda en áreas más pequeñas y calcular la pendiente media en cada una de estas nuevas regiones. Este sensor se compone de una red de micro lentes, también conocidas como subaperturas, todas del mismo tamaño. Cada una de estas subaperturas se proyecta en un sensor CCD, que asigna una cantidad idéntica de píxeles a cada subapertura. Esto resulta especialmente útil, ya que la imagen de la estrella obtenida a través del sensor coincide con la configuración de los espejos deformables utilizados para corregir las aberraciones.

3.1.2.2. Estrellas guía

Al diseñar un sistema de óptica adaptativa, uno de los componentes clave a considerar es el sensor de frente de onda presentado en el apartado anterior. Este sensor emplea las denominadas estrellas guía. Estas estrellas pueden ser naturales, ubicadas en el campo de visión del objeto de interés y previamente caracterizadas, o artificiales, generadas mediante un láser disparado desde tierra que refleja en las capas altas de la atmósfera y es observable desde el telescopio.

Al implementar sistemas de óptica adaptativa en telescopios, las estrellas guía naturales son esenciales. Estas estrellas, que son bien conocidas y completamente caracterizadas, se usan como referencia para corregir la turbulencia atmosférica. En una configuración simple con un solo espejo deformable y un sensor, la situación ideal es tener una estrella guía natural en el mismo eje de observación que el objeto a observar. En este caso, la corrección de la turbulencia solo depende de la velocidad del sistema y la resolución del espejo.

Otra configuración importante es la óptica adaptativa para múltiples objetos, utilizada en instrumentos como el MOSAIC del ELT. Aquí, cada objeto astronómico observado tiene su propio espejo deformable, lo que permite corregir la turbulencia en áreas pequeñas alrededor de cada objeto. Esta técnica emplea estrellas guía naturales cercanas a cada objeto como referencia, permitiendo observar múltiples objetivos en un campo amplio y mejorando la calidad de la observación mediante correcciones individuales.

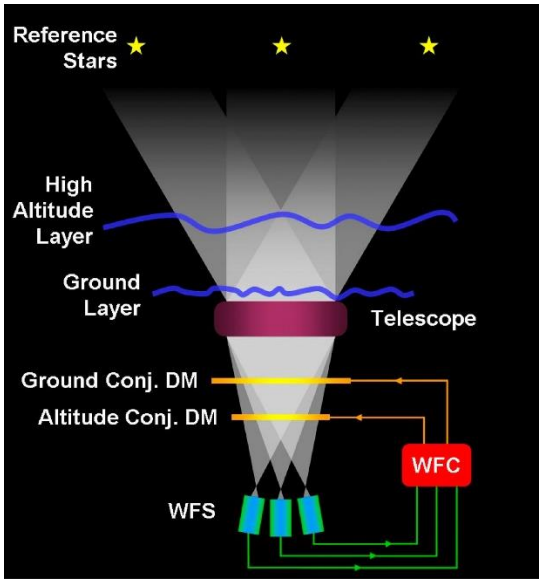

Figura 3.2- Representación gráfica de una configuración *Multi-object adaptive optic* con dos estrellas de referencia.

Por otro lado, las estrellas guía láser (LGS, por sus siglas en inglés) son componentes esenciales en los sistemas de óptica adaptativa utilizados en telescopios terrestres, especialmente cuando las estrellas naturales no son adecuadas o están ausentes en la proximidad del objeto de observación.

Estas estrellas artificiales se crean proyectando un haz de láser hacia la mesosfera, creando un punto de luz que puede ser utilizado como referencia para medir y corregir las distorsiones atmosféricas. Existen diferentes tipos de estrellas guía artificiales; algunas se generan a altitudes de 80-90 km, utilizando la excitación de átomos de sodio, mientras que otras se lanzan a altitudes más bajas, alrededor de 20 km, aprovechando la dispersión de Rayleigh en las regiones más densas de la atmósfera [4].

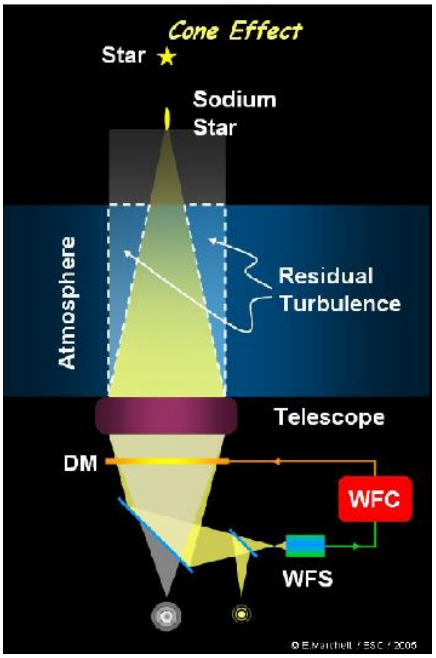

Figura 3.3- Representación del efecto cono producido en una estrella artificial.

A pesar de su utilidad, las LGS tienen limitaciones como su incapacidad para medir el desplazamiento total del frente de onda debido al llamado "efecto cono", el cual como se puede observar en la Figura 3.4. Este es una discrepancia geométrica que ocurre porque la luz de la LGS se dispersa en forma de cono mientras que la luz de la estrella observada viaja en una columna recta. No obstante, su implementación ha significado un avance sustancial en la capacidad de los telescopios terrestres para obtener imágenes claras y detalladas del espacio, superando significativamente las restricciones impuestas por la disponibilidad de estrellas guía naturales [5].

3.1.2.3. Reconstructor tomográfico

El reconstructor tomográfico es un componente crítico en los sistemas de óptica adaptativa puesto que interpreta los datos proporcionados por el sensor de frente de onda para determinar las correcciones que el espejo deformable debe realizar. Utilizando algoritmos matemáticos avanzados, el reconstructor analiza las aberraciones detectadas y calcula los valores óptimos para ajustar el espejo deformable de manera que se compense la distorsión óptica causada por la atmósfera. Esta unidad puede emplear diversas técnicas

de reconstrucción, desde métodos lineales basados en mínimos cuadrados hasta enfoques más complejos como redes neuronales artificiales, dependiendo de la precisión y velocidad requeridas.

Se puede afirmar que el reconstructor interpreta las aberraciones detectadas por el sensor de frente de onda y calcula los ajustes necesarios para el espejo deformable para corregir estas distorsiones en tiempo real [4].

La eficacia del reconstructor es fundamental para maximizar la capacidad del sistema de óptica adaptativa para mejorar la calidad de las imágenes astronómicas, permitiendo observaciones más claras y detalladas de los objetos celestes, tal y como podemos ver en la Figura 3.5.

Figura 3.4- Imagen de Neptuno con y sin óptica Adaptativa captadas desde el Very Large Telescope (VLT) de ESO.

3.1.2.4. Espejo deformable

Con la información obtenida previamente, la misión de estos espejos será cambiar su forma para conseguir así solucionar la deformación causada por las turbulencias atmosféricas captadas en las observaciones. Estos espejos, que son flexibles, modificarán

su superficie de tal manera que se pueda adaptar al frente de onda captado con los diferentes sensores, para así conseguir el objetivo de relejar un frente de onda lo más plano posible.

La producción de estos espejos siempre ha representado uno de los desafíos más grandes del proceso, puesto que se basa en fijar actuadores a carcasas ópticas delgadas de gran diámetro para finalmente trasladarlas a lugares generalmente alejados de zonas habitadas. Cabe destacar que un actuador es un dispositivo que recibe una entrada de energía y la convierte en movimiento o fuerza.

En los últimos años se han implementado sistemas que combinan funciones electrónicas, mecánicas y ópticas en componentes pequeños, por lo que la fabricación de estos ha mejorado con creces [6].

Los espejos deformables no solo representan una solución eficaz a las turbulencias atmosféricas, sino que también son un ejemplo de cómo la innovación y la tecnología pueden colaborar para superar desafíos complejos en la observación astronómica.

3.2. REDES NEURONALES Y SU APLICACIÓN A LA RECONSTRUCCION ÓPTICA

La utilización de redes neuronales artificiales representa una intersección entre la biología y la tecnología, puesto que se inspiran en la estructura y funcionamiento del cerebro humano, como se puede observar en la Figura 3.5.

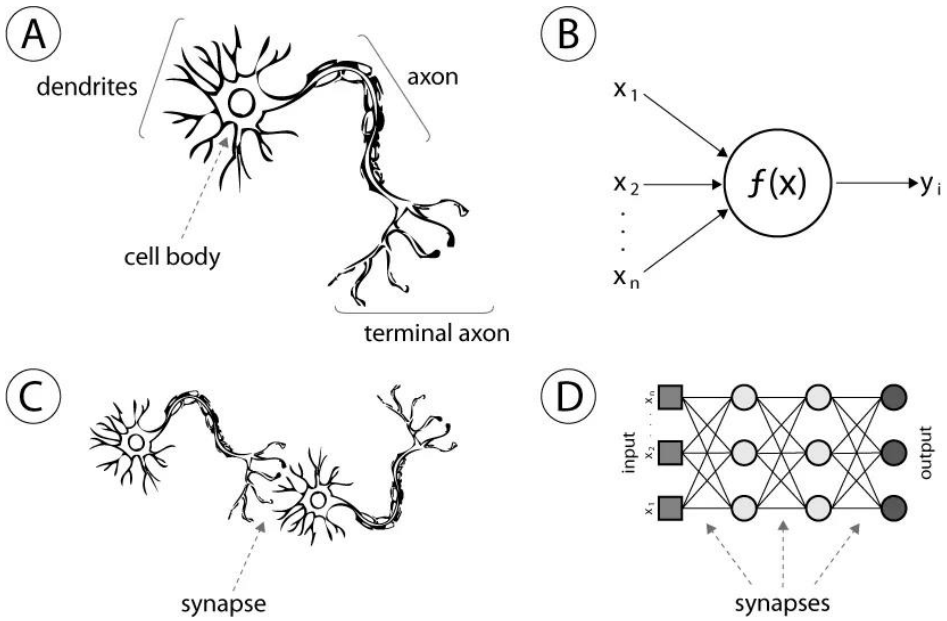

Figura 3.5- Comparación de una red neuronal biológica y una artificial: A) Representación de una neurona biológica, B) Modelo de neurona artificial, C) Sinapsis entre dos neuronas biológicas y D) Red neuronal artificial).

Estas estructuras computacionales imitan el proceso de aprendizaje del cerebro a través de un sistema de nodos y conexiones que emulan a las neuronas y sus sinapsis. Las sinapsis son básicamente entradas ponderadas, lo que significa que se puede tomar una entrada y multiplicarla por un peso específico para dicha entrada.

Esta capacidad para aprender, ajustando los diferentes pesos y adaptarse ha hecho que las redes neuronales artificiales sean herramientas extremadamente útiles en una variedad de campos, incluyendo el de la óptica adaptativa.

En el campo de la reconstrucción tomográfica, las redes neuronales artificiales se aplican para mejorar la calidad de las imágenes capturadas bajo condiciones adversas de observación, tales como turbulencias atmosféricas o imperfecciones en los sistemas de captura de imágenes. Se comenzaron a utilizar desde la década de los 90, aunque la integración de técnicas más avanzadas de aprendizaje automático y redes neuronales es relativamente reciente [7].

3.2.1. Redes Neuronales Artificiales

Las redes neuronales artificiales (ANNs) tienen sus raíces en los esfuerzos por entender y replicar los procesos del cerebro humano para resolver problemas complejos del mundo real. Aunque los conceptos básicos de las ANNs se remontan a ideas de principios del siglo XX, fue el trabajo de McCulloh y Pitts en 1943 el que estableció un modelo matemático de neuronas, demostrando que simples redes neuronales podrían calcular funciones aritméticas y lógicas [8].

Durante los años 50 y 60, este campo ganó un impulso significativo al que se sumó la introducción del Perceptrón por parte de Frank Rosenblatt [9]. Este modelo era capaz de realizar tareas de reconocimiento de patrones mediante un proceso de aprendizaje simple. A pesar de sus limitaciones, que solo se hicieron evidentes más tarde con las críticas de Minsky y Papert, el perceptrón sentó las bases para los desarrollos futuros en la computación neuronal.

La investigación en ANNs experimentó un período de estancamiento debido a estas críticas, pero resurgió con fuerza en los años 80 gracias al algoritmo de retropropagación que permitió el entrenamiento de redes multicapas, abriendo nuevas posibilidades para el aprendizaje y la representación de conocimientos complejos. Este renacimiento también estuvo marcado por el uso creciente de la teoría del aprendizaje y técnicas estadísticas que ayudaron a mejorar el diseño y la eficacia de las ANNs.

La historia de las redes neuronales convolucionales y su impacto en la clasificación de imágenes ha sido revolucionaria, especialmente con la incorporación de las Unidadades de Procesamiento Gráfico (GPUs) y técnicas avanzadas de entrenamiento. A partir del trabajo seminal de Krizhevsky, Sutskever y Hinton en 2012, se demostró el poder de las CNNs para clasificar imágenes a gran escala. Esto se evidenció en su notable desempeño en la competencia ImageNet LSVRC-2010 [10].

La implementación eficiente de operaciones convolucionales en GPUs permitió que estas redes grandes y complejas se entrenaran en un tiempo razonable, aprovechando al máximo las capacidades de procesamiento paralelo de las GPUs.

En los años siguientes, la utilización de GPUs se ha convertido en un estándar en la investigación de redes neuronales profundas, permitiendo el entrenamiento de modelos aún más grandes y complejos. Las redes convolucionales han evolucionado para incluir arquitecturas más profundas y sofisticadas, como ResNet y Inception, que han continuado mejorando el rendimiento en diversas tareas de visión por computadora. Estas tecnologías, impulsadas por la capacidad de procesamiento masivo de las GPUs y las técnicas de optimización avanzadas, han ampliado los límites de lo que es posible en el reconocimiento y clasificación de imágenes, llevando a aplicaciones prácticas en múltiples campos, desde la medicina hasta la conducción autónoma [11].

3.2.1.1. Aprendizaje en redes neuronales

En el campo de la óptica adaptativa, las redes neuronales artificiales juegan un papel crucial en la mejora de las técnicas de reconstrucción óptica [12]. Un componente fundamental de estas redes es el perceptrón multicapa, del cual se hablará en profundidad más delante, que se organiza en varias capas de neuronas interconectadas donde cada neurona en una capa se conecta con todas las neuronas en la capa siguiente, facilitando un flujo estructurado de información.

Este tipo de red es capaz de procesar y aprender de los datos de entrada a través de un método conocido como aprendizaje supervisado, donde el sistema se entrena utilizando conjuntos de datos que incluyen entradas y salidas deseadas.

El proceso de aprendizaje de una red neuronal involucra múltiples etapas, comenzando con la propagación hacia adelante de los datos a través de la red para generar una salida. Esta salida se compara con la salida deseada y la diferencia entre ambas se utiliza para calcular un error, que se utiliza posteriormente para ajustar los pesos de las conexiones neuronales mediante un proceso conocido como retropropagación.

A través de repetidas iteraciones o 'epochs', la red ajusta sus pesos para minimizar el error en sus predicciones, afinando su capacidad para reconstruir con precisión las aberraciones ópticas observadas.

En el proceso de entrenamiento de una red neuronal, varios componentes son fundamentales, siendo uno de los más críticos la función de coste. Esta función evalúa la predicción o el funcionamiento de la red al compararla con el valor real o esperado en dicha salida.

Aunque la función de coste puede variar según el problema específico, el error más utilizado para problemas de regresión suele ser el Error Cuadrático Medio (MSE, por sus siglas en inglés). La fórmula del MSE se define como sigue:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \qquad (4)$$

Donde $\hat{y}$ representa la salida predicha por la red, y es el valor real de la salida, y n es el número total de muestras en el conjunto de datos. Una función de coste baja indica un error pequeño, lo que significa que las predicciones de la red son cercanas a los valores reales. Durante el entrenamiento, el objetivo es ajustar los pesos w y los sesgos b (pertenecientes a la ecuación 13, del apartado 3.2.1.2, la cual representa como procesa la entrada en una red neuronal) de la red para minimizar el MSE entre la salida esperada y las predicciones. Este proceso asegura que la red neuronal refine progresivamente sus predicciones hacia las salidas reales, optimizando su rendimiento al aproximar con precisión los valores objetivo en cada par de entrada-salida posible.

Una posible solución para determinar el mínimo global de la función de coste en redes neuronales sería resolverla analíticamente. No obstante, dado que las funciones de coste en redes neuronales dependen de un gran número de variables, que corresponden al número de neuronas en la red, resulta extremadamente costoso computacionalmente resolverla mediante el cálculo de segundas derivadas parciales debido al enorme volumen de operaciones requeridas, que excede la capacidad de cálculo disponible en los ordenadores actuales.

Como alternativa, se propone el uso del algoritmo de descenso del gradiente para encontrar un mínimo de la función de coste. Mediante este método, se recorre el espacio n-dimensional de la función de coste mediante pequeños pasos iterativos, buscando un mínimo que se aproxime lo más posible a la solución óptima. Aunque el descenso del gradiente no garantiza la localización del mínimo global, permite encontrar un mínimo local de calidad suficiente como para considerar detener la búsqueda. La explicación

detallada de este proceso se basa en la manipulación de n variables a y la función de coste $C(a)$, empleando aproximaciones matemáticas específicas [13]. Las cuales son:

$$\Delta C \approx \frac{\partial C}{\partial a_1}\Delta a_1 + \frac{\partial C}{\partial a_2}\Delta a_2 + \cdots + \frac{\partial C}{\partial a_n}\Delta a_n \quad (5)$$

El objetivo, teniendo en cuenta la ecuación anterior, sería obtener la suma que haga que ΔC tienda a 0, dado que esto significaría que el valor cada vez es más pequeño y por lo tanto se está minimizando el coste.

El gradiente de la función de coste puede definirse de la siguiente forma:

$$\nabla_a C \equiv \left(\frac{\partial C}{\partial a_1}, \frac{\partial C}{\partial a_2}, \ldots, \frac{\partial C}{\partial a_n}\right)^T \quad (6)$$

Permitiendo también reescribir la ecuación anterior:

$$\Delta C \approx \nabla_a C \cdot \Delta a \quad (7)$$

Esto revela una conexión aún más directa entre las variaciones en las variables a y los cambios en la función de coste. Es factible elegir las variaciones en a de tal manera que se minimice el valor de $\nabla_a C$, lo cual se consigue con el valor:

$$\Delta a = -\lambda \nabla_a C \quad (8)$$

Donde λ es siempre un valor positivo conocido como la tasa de aprendizaje. Este parámetro es muy importante en el proceso de entrenamiento ya que influye en la velocidad de actualización de la función de coste.

Al unificar las ecuaciones se obtiene:

$$\Delta C \approx -\lambda \nabla_a C \cdot \nabla_a C = -\lambda \|\nabla_a C\|^2 \quad (9)$$

De aquí se deduce que el valor de la fórmula anterior siempre será negativo, ya que depende de λ (positivo por definición) y el cuadrado del gradiente de C (siempre positivo). Por lo tanto, en cada iteración, a adoptará un nuevo valor de la siguiente forma:

$$a' = a - \lambda \nabla C \quad (10)$$

Con suficientes iteraciones, el algoritmo convergerá a un mínimo local de la función de coste. Como se ha comentado anteriormente, aunque esto no garantiza una solución óptima, en la práctica es posible obtener un mínimo local bastante adecuado al repetir el proceso con diferentes valores iniciales de a.

Este proceso puede aplicarse al entrenamiento de una red neuronal, permitiendo modificar los pesos y sesgos de cada neurona para minimizar el error en las predicciones. Calcular el gradiente de C con respecto a *a* es directo cuando se dispone de una función de coste definida específicamente, facilitando su derivada.

Si se considera que *a* es una función de *w* y *b* (pesos y sesgos), la actualización de estos valores seguirá un proceso similar, asegurando una mejora continua en la precisión de la red.

En las redes neuronales, la actualización de pesos y sesgos en cada capa implica desafíos particulares, ya que las capas ocultas (aquellas que no son de salida ni entrada) no poseen una función de coste directa para derivar sus gradientes. La técnica de retropropagación aborda esta dificultad al permitir que el gradiente del error se propague hacia atrás desde la salida hasta la entrada, actualizando los pesos en cada capa de manera efectiva.

Se puede considerar que a^L representa los valores de una neurona en la capa de salida, y a^{L-1} los valores de salida de la capa anterior. La relación entre estas activaciones se puede expresar como:

$$a^L = \sum_{i=0}^{n}(w_i \cdot a^{L-1}) + b \quad (11)$$

Donde w_i son los pesos asociados a cada conexión, y b es el sesgo.

Para facilitar el cálculo de gradientes respecto a las activaciones de capas anteriores, representamos a^{L-1} en función de a^L. Esto permite aplicar la regla de la cadena en la derivación parcial del gradiente de la función de coste respecto a las activaciones de la capa anterior:

$$\nabla_{a^{L-1}}C = \frac{\partial C}{\partial a^L} \cdot \frac{\partial a^L}{\partial a^{L-1}} \quad (12)$$

Al desglosar los términos se observa que la primera derivación refleja cómo el cambio en la capa de salida afecta al coste, mientras que la segunda derivación implica ajustar los pesos que conectan las capas. Este proceso se replica a través de todas las capas de la red, permitiendo así la actualización eficaz de todos los pesos y sesgos en función del error calculado en la salida.

La implementación de este proceso debe considerar varios factores críticos no automáticos, como la frecuencia de actualización de los pesos. Típicamente, el proceso se realiza con un par de datos de entrada-salida a la vez, lo que puede ser subóptimo ya que cada par de datos buscará minimizar el error para ese caso específico y no para el conjunto global de datos.

Además, este enfoque puede resultar en un proceso computacionalmente intensivo y lento debido a que las multiplicaciones matriciales, especialmente en hardware no optimizado, no utilizan plenamente la capacidad computacional disponible.

El entrenamiento de redes neuronales se puede optimizar significativamente mediante el uso de técnicas avanzadas. El descenso de gradiente estocástico (SGD), especialmente su variante que utiliza mini-lotes, es fundamental para realizar actualizaciones frecuentes de los pesos utilizando subconjuntos aleatorios de datos. Esto facilita la búsqueda del mínimo global de la función de coste de manera más eficiente y efectiva.

Adicionalmente, la técnica de *momentum* permite acelerar la convergencia del proceso hacia el mínimo de la función de error, especialmente cuando el proceso está lejos de este mínimo. Este método ajusta la magnitud de las actualizaciones basadas en su posición relativa al mínimo deseado, lo que mejora la eficacia del entrenamiento.

Es importante también considerar el valor inicial de los pesos y los sesgos, así como la selección adecuada de los datos de entrenamiento. Estos factores son determinantes en la calidad del entrenamiento y en cómo la red logrará generalizar después del mismo. Un conjunto de datos bien seleccionado y variado amplía la capacidad de generalización de la red, aunque esto también introduce la necesidad de balancear la cantidad de datos utilizados con la capacidad de procesamiento y el tiempo disponible para el entrenamiento [14].

3.2.1.2. Perceptrón Multicapa

Las redes neuronales consisten en conjuntos de neuronas interconectadas que permiten que la salida de una neurona influya en la entrada de otra. En estas redes, cada neurona puede recibir múltiples entradas y la cantidad de estas depende del número de

conexiones que tiene con otras neuronas. Cada entrada es multiplicada por un peso específico, lo que permite ponderar su relevancia dentro del cálculo global. Tras ponderar las entradas, se suman y se les añade un término conocido como sesgo o *bias*. A este resultado se le aplica una función de activación, que puede variar entre tipos como la sigmoide, la tangente hiperbólica o la función rectificadora (ReLU) entre otros [15].

Este proceso se describe matemáticamente mediante la siguiente ecuación, la cual resume cómo se procesa la entrada en una red neuronal:

$$Y = f(\sum_{i=0}^{n}(w_i \cdot x_i) + b) \quad (13)$$

Donde x_i representa las entradas a la neurona, w_i son los pesos que modulan la importancia de cada entrada, y b es el sesgo que permite ajustar la salida del modelo además de la suma ponderada de las entradas. La función f es la función de activación que introduce no linealidades en el modelo, permitiendo a la red aprender y modelar relaciones complejas. Estos componentes trabajan en conjunto para transformar la entrada en una salida que la red utiliza para realizar tareas como clasificación o regresión.

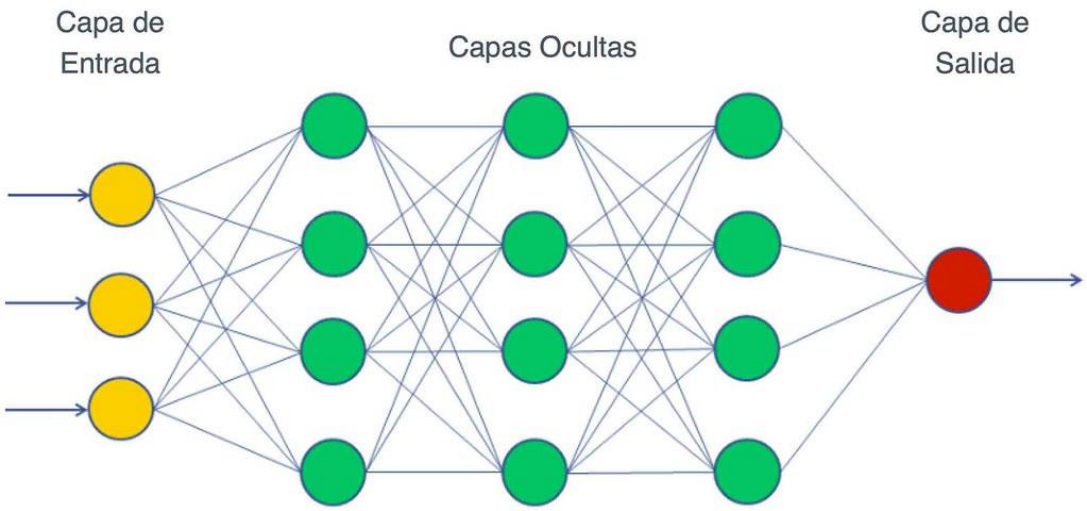

Figura 3.6- Representación del perceptrón multicapa, donde las flechas indican las conexiones entre neuronas.

En la Figura 3.6 se muestra la arquitectura típica de un perceptrón multicapa, que incluye una capa de entrada, varias capas ocultas y una capa de salida. Las neuronas de

cada capa están completamente conectadas con las de la capa siguiente, y no existen conexiones entre neuronas de la misma capa ni retroalimentaciones. Este tipo de red es capaz de aprender y modelar relaciones no lineales complejas.

La configuración de parámetros en una red neuronal es fundamental para su eficacia operativa. Estos parámetros no solo influyen en la propagación hacia adelante durante el entrenamiento, sino también en cómo se optimizan los pesos y los sesgos a lo largo del tiempo. Factores como la cantidad de neuronas en cada capa y la elección de la función de activación afectan directamente tanto la calidad del aprendizaje como la velocidad de este. Durante la inferencia, el número adecuado de neuronas en cada capa es esencial para manejar adecuadamente la complejidad de los datos y evitar el sobredimensionamiento o subdimensionamiento de la red. Además, la función de activación seleccionada puede afectar significativamente cómo la red aprende y generaliza a partir de los datos de entrenamiento.

La modificación del número de capas, la tasa de aprendizaje y la introducción de técnicas como el *dropout* pueden influir en el rendimiento de modelos de red neuronal profunda.

Estas consideraciones son esenciales para diseñar redes neuronales que no solo sean eficaces en términos de aprendizaje y memoria, sino también en términos de eficiencia computacional y capacidad de generalización [16].

El *dropout* es una técnica de regularización que se utiliza para prevenir el sobreajuste en redes neuronales profundas. Fue introducida por *Srivastava et al. en 2014*[17] y consiste en desactivar de manera aleatoria una fracción de las neuronas durante el entrenamiento. Específicamente, en cada iteración del entrenamiento, cada neurona se "descarta" con una probabilidad p (por ejemplo, 0.5), es decir, su activación se establece en cero y no se considera en la pasada hacia adelante ni en la retropropagación.

La normalización de los valores de salida en una red neuronal es una técnica para mejorar la eficacia del aprendizaje y la generalización. Al ajustar los valores de salida para que se adapten a un rango limitado, se facilita el uso de diversas funciones de activación. Esta normalización no solo es crucial para la capa de salida, sino que también debe

considerarse para las capas ocultas, donde se pueden utilizar funciones de activación completamente diferentes.

Además, la elección de la cantidad de capas ocultas y el número de neuronas en cada una de estas influye en la capacidad de la red para modelar funciones complejas y, por ende, para abordar problemas más sofisticados. Un mayor número de capas y neuronas generalmente permite una mejor aproximación de funciones complejas, lo que a su vez mejora la capacidad de la red para generalizar y resolver problemas más complicados. Sin embargo, esta expansión en la complejidad de la red puede tener un impacto negativo en el proceso de entrenamiento. A medida que aumenta el número de parámetros a ajustar, debido a un mayor número de neuronas y capas, el proceso de aprendizaje se vuelve más desafiante y computacionalmente costoso.

Este fenómeno es conocido en la literatura como el compromiso entre la capacidad de la red y su eficiencia de entrenamiento. Aunque redes más grandes pueden teóricamente modelar relaciones más complejas, requieren ajustes más cuidadosos y métodos de regularización efectivos para evitar el sobreajuste y garantizar una buena generalización en datos no vistos durante el entrenamiento. Este análisis resalta la importancia de un diseño cuidadoso de la arquitectura de la red, donde se balanceen la capacidad y la eficiencia para alcanzar un rendimiento óptimo [16].

3.2.1.3. Redes neuronales convolucionales

Las Redes neuronales convolucionales son una categoría especializada de redes neuronales que han revolucionado el campo de la visión por computadora y el análisis de imágenes. A diferencia de las redes neuronales tradicionales, que procesan los datos de entrada en una forma completamente conectada, las CNN implementan una estructura que imita la percepción visual humana. Esta estructura permite que las CNN detecten patrones complejos en imágenes a través de múltiples capas de procesamiento, cada una de las cuales aprende a identificar características desde las más simples hasta las más complejas.

Una característica distintiva de las CNN es su capacidad para preservar la relación espacial entre los píxeles al aplicar filtros convolucionales, lo que las hace particularmente potentes para tareas como la clasificación de imágenes, la detección de objetos y la

segmentación. Estos filtros, también conocidos como *kernels*, se desplazan sobre la imagen de entrada para producir mapas de características que resumen la presencia de patrones específicos en diferentes ubicaciones de la imagen. Esta operación no solo reduce la dimensionalidad de los datos, sino que también permite que la red se enfoque en las características más relevantes.

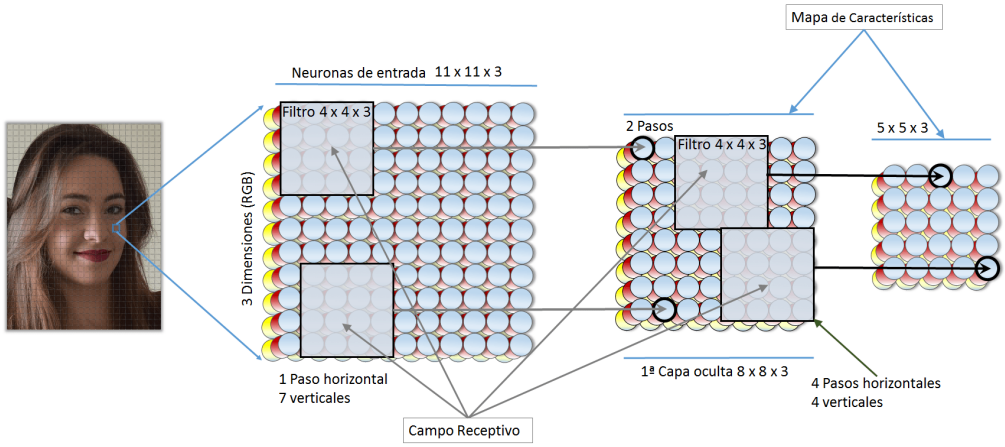

Figura 3.7- Visualización del procesamiento de redes neuronales convolucionales en la clasificación y detección de imágenes.

Las CNN se estructuran de manera que las capas iniciales y las capas de *pooling* trabajan conjuntamente para identificar y condensar patrones útiles dentro de los datos visuales. Las capas de *pooling*, también conocidas como capas de agrupamiento, reducen progresivamente la dimensionalidad de la representación de los datos, lo que disminuye el número de parámetros y la carga computacional de la red. Existen varios tipos de *pooling*, siendo el *max-pooling* uno de los más comunes, que selecciona el valor máximo de una ventana de filtrado y lo utiliza para representar esa región.

Según se destaca en la literatura, estas capas están diseñadas para extraer características progresivamente más abstractas a medida que la información fluye a través de la red [18]. Posteriormente, las capas completamente conectadas utilizan estas características resumidas para hacer predicciones o clasificaciones. Este diseño permite que las CNN no solo procesen imágenes de manera efectiva, sino que también operen con una notable rapidez y eficiencia, manejando grandes volúmenes de datos, lo que las hace cruciales en aplicaciones de aprendizaje profundo y visión por computadora.

Las CNN combinan las convoluciones con capas totalmente conectadas, ofreciendo ventajas significativas sobre la transformación de imágenes de entrada en un vector unidimensional para el uso exclusivo de perceptrones multicapa de principio a fin.

En las CNN, las convoluciones reducen drásticamente el número de parámetros y operaciones necesarias en comparación con las capas totalmente conectadas. En lugar de enlazar cada píxel de la imagen con todos los valores de la siguiente capa, se utiliza un filtro o kernel que se desplaza sobre la imagen, realizando operaciones de convolución. Esta metodología es eficaz cuando la información relevante se encuentra en los vecinos inmediatos de cada entrada, como es común en el procesamiento de imágenes.

Sin embargo, existe una consideración importante: aunque el número de operaciones y parámetros en las capas convolucionales es menor, el tamaño del vector resultante tras varias convoluciones y su conversión a una dimensión puede ser significativamente mayor que el tamaño de la imagen original. Esto podría aumentar el tamaño de las capas totalmente conectadas más allá de lo que se obtendría si no se hubiesen utilizado convoluciones. Para mitigar este efecto, se propone reducir el tamaño de la imagen después de cada convolución. El método más común para esta reducción es el *max pooling*, donde se seleccionan regiones de la imagen de NxN píxeles y se reduce a un solo valor, el cual será el máximo encontrado en esa región. En la Figura 3.8 podemos ver como se aplica un *max pooling* de 2x2 y en consecuencia se reduce la salida a un cuarto de la imagen original.

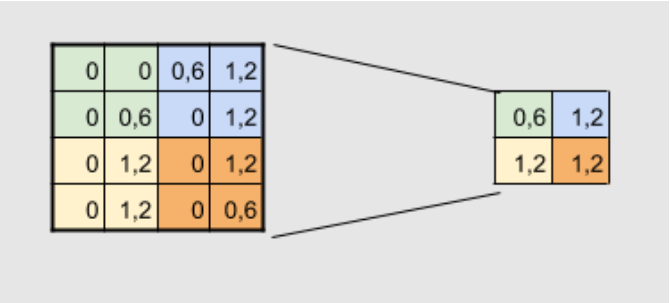

Figura 3.8- Ejemplo de implementación de *Max Pooling* para reducción dimensional en CNN.

3.2.2. Computación paralela en tarjetas gráficas

En el ámbito del avance computacional de las últimas décadas, uno de los logros más significativos ha sido el aumento exponencial en la capacidad de procesamiento, lo cual ha hecho viable la ejecución de operaciones y simulaciones extremadamente complejas. Tradicionalmente, la unidad de procesamiento central (CPU) ha sido el corazón de los sistemas informáticos, encargada de ejecutar las instrucciones de los programas mediante uno o varios núcleos que, junto a la frecuencia de reloj, determinan la cantidad de operaciones que pueden procesarse por segundo.

Sin embargo, en el contexto actual de la informática, las tarjetas gráficas o unidades de procesamiento gráfico han ganado un papel protagonista, extendiendo su uso más allá de la simple renderización gráfica para convertirse en una pieza clave en la computación paralela. Las GPUs están especialmente diseñadas para manejar múltiples operaciones de manera simultánea, lo que las hace ideales para tareas que requieren un alto grado de paralelismo, como es el caso de la simulación de fenómenos físicos, la minería de datos, y especialmente, el aprendizaje profundo y las redes neuronales.

La evolución de las GPUs hacia la computación de propósito general (GPGPU) permite que estos componentes no solo gestionen gráficos, sino que también ejecuten algoritmos que tradicionalmente se realizaban en CPUs. Esto ha revolucionado campos como la inteligencia artificial, donde las capacidades de procesamiento paralelo de las unidades de procesamiento gráfico facilitan operaciones intensivas como el entrenamiento de modelos de datos grandes y complejos.

A lo largo de los siguientes apartados, se explorará en detalle la computación paralela en las tarjetas gráficas, la computación de propósito general en GPUs, la paralelización de operaciones y, finalmente, cómo estas están siendo utilizadas específicamente en redes neuronales, destacando su impacto y las ventajas que ofrecen en estos campos.

3.2.2.1. Computación de propósito general en tarjetas gráficas

Las GPU están diseñadas con una arquitectura que favorece una gran cantidad de núcleos pequeños y especializados para tareas específicas, a diferencia de las CPU que generalmente poseen menos núcleos, pero con una capacidad de procesamiento más diversificada. Esta configuración permite a las GPU prestaciones excelentes en tareas que requieren el manejo simultáneo de numerosas operaciones, como el procesamiento de imágenes y la ejecución de algoritmos de aprendizaje automático.

Una diferencia fundamental entre GPU y CPU radica en su manejo de la memoria. Las GPU utilizan VRAM, optimizada para gráficos y procesos intensivos de datos, mientras que las CPU dependen de la RAM estándar que sirve a un espectro más amplio de aplicaciones. El uso de VRAM permite a las GPU gestionar eficientemente grandes bloques de datos gráficos, lo que es crucial para aplicaciones de renderizado y simulación en tiempo real.

Figura 3.9- Comparación arquitectónica y de manejo de memoria entre CPU y GPU.

Las GPU no solo se comunican de forma efectiva con las CPU y utilizan RAM del sistema, sino que también están configuradas para trabajar en conjuntos, permitiendo que múltiples tarjetas gráficas operen en paralelo para incrementar significativamente la capacidad de procesamiento. Esta capacidad para escalar hace a las GPU particularmente valiosas en entornos de investigación y aplicaciones industriales donde el tiempo de procesamiento y la capacidad de manejo de datos son críticos [19].

En la Figura 3.9, se ilustra una comparación entre la arquitectura de una CPU y una GPU. Como se puede observar, las CPU tienen un diseño con varias unidades de procesamiento aritmeticológico (ALU) controladas centralmente y una caché significativa para gestionar las operaciones. En contraste, las GPU están compuestas por una gran cantidad de ALUs, lo que les permite realizar muchas operaciones en paralelo, pero con una menor capacidad de control central y caché.

Además de la capacidad de instalar múltiples GPUs en un solo sistema, lo cual amplía el número de núcleos disponibles para procesamiento paralelo, es importante considerar los desafíos técnicos asociados con esta configuración. Aunque agregar más GPUs incrementa los núcleos, también complica la gestión de la comunicación entre ellas y con la RAM principal. Todas las GPUs comparten el mismo bus PCIe, necesario tanto para la comunicación interna como con la RAM. Esta configuración puede limitar las ganancias en velocidad de procesamiento que de otro modo serían posibles debido a cuellos de botella en la transferencia de datos.

3.2.2.2. Paralelización de operaciones

La paralelización en computación implica descomponer tareas complejas en subprocesos más pequeños que se ejecutan simultáneamente, aprovechando la arquitectura de las GPUs, diseñadas específicamente para este propósito.

Estas unidades cuentan con múltiples núcleos que permiten procesar diversas operaciones al mismo tiempo, optimizando así el rendimiento en aplicaciones intensivas como análisis de datos o simulaciones. Este enfoque no solo acelera los cálculos, sino que

también maximiza la eficiencia del hardware, haciendo las GPUs herramientas esenciales en el campo de la computación de alto rendimiento.

Modelos de programación como CUDA (Compute Unified Device Architecture) [20] de NVIDIA y OpenCL (Open Computing Language) [21] son fundamentales en este proceso, ya que proporcionan los marcos y herramientas necesarios para desarrollar aplicaciones que aprovechan la paralelización en GPUs.

CUDA y OpenCL permiten a los desarrolladores asignar tareas específicas a los núcleos de las GPUs, donde operaciones frecuentes como la multiplicación de matrices y la convolución se realizan con una eficiencia significativamente mayor en comparación con las CPUs tradicionales. La multiplicación de matrices es esencial en muchos algoritmos de aprendizaje automático y simulaciones físicas, mientras que la convolución es una operación clave en el procesamiento de señales e imágenes, fundamentales en áreas como la visión por computadora y el análisis de datos. Exploraremos estos temas en mayor profundidad en los siguientes puntos [22].

La paralelización ofrece múltiples ventajas significativas en la computación, especialmente en el uso de GPUs:

- **Eficiencia en el procesamiento**: La capacidad de ejecutar múltiples cálculos simultáneamente permite a las GPUs manejar grandes volúmenes de datos mucho más rápido que las CPUs convencionales lo que es muy importante para aplicaciones que involucran grandes conjuntos de datos y cálculos complejos.

- **Reducción del tiempo de procesamiento**: La paralelización reduce drásticamente el tiempo necesario para realizar operaciones intensivas. Esto se traduce en un aumento de la productividad y la capacidad para realizar más análisis en menos tiempo.

- **Escalabilidad**: Con la paralelización, es posible escalar las operaciones de procesamiento según sea necesario, lo que permite a las organizaciones ajustar los recursos de procesamiento para satisfacer la demanda sin degradar el rendimiento.

- **Eficiencia energética**: Al maximizar la capacidad de procesamiento de cada GPU, se puede lograr una mayor eficiencia energética en comparación con los sistemas que dependen exclusivamente de CPUs.

3.2.2.3. Utilización de GPUs en redes neuronales

En el campo de la inteligencia artificial, y más concretamente en el aprendizaje profundo, la utilización de unidades de procesamiento gráfico ha revolucionado la manera en que se entrenan los modelos de redes neuronales. *Frameworks* como TensorFlow [23] han sido diseñados para aprovechar la computación paralela que ofrecen las GPUs, permitiendo una reducción significativa en los tiempos de entrenamiento y una mejora en la eficiencia del procesamiento de grandes volúmenes de datos.

TensorFlow, desarrollado por Google, utiliza una arquitectura que facilita la construcción y el entrenamiento de modelos de aprendizaje profundo mediante la definición de operaciones en forma de grafos de flujo de datos. Estos grafos se ejecutan de manera óptima en GPUs, aprovechando su capacidad para realizar múltiples operaciones de forma simultánea. Esta capacidad es especialmente crítica para operaciones como la multiplicación de matrices y la convolución, fundamentales en muchos tipos de redes neuronales [24].

Para comprender mejor el impacto de las GPUs en el rendimiento de los frameworks de aprendizaje profundo, podemos referirnos a investigaciones que comparan directamente el rendimiento de TensorFlow en configuraciones de CPU y GPU, evidenciando las ventajas de la paralelización en GPUs. Estos estudios destacan cómo las operaciones que antes tomaban horas o días, ahora se pueden completar en minutos o segundos, democratizando el uso de tecnologías de aprendizaje profundo y permitiendo experimentación y desarrollo más rápidos.

Uno de estos estudios es el comparativo realizado por *DATAmadness*, el cual evidenció que al entrenar un modelo de red neuronal convolucional con TensorFlow, el tiempo de entrenamiento en una CPU fue aproximadamente de 480 segundos por época,

mientras que el mismo entrenamiento en una GPU tomó solo 75 segundos por época, lo que representa una reducción del 85% en el tiempo de entrenamiento. Durante el uso de la GPU, se observó una utilización del 11% de la GPU con toda la memoria de 8GB utilizada, en contraste con el uso del CPU, que mostró una utilización del 80%.

Además de TensorFlow, otros frameworks como PyTorch y Keras también se benefician de la aceleración por GPU, cada uno ofreciendo diferentes enfoques y sintaxis para aprovechar esta tecnología. La elección del *framework* puede depender de la especificidad del proyecto, la preferencia del desarrollador y la comunidad de apoyo disponible, pero todos comparten la ventaja común de la integración optimizada con las GPUs [24].

3.2.3. Integración en otras ramas

Las redes neuronales y la inteligencia artificial (IA) han experimentado una notable expansión, transformando industrias y campos de estudio con su capacidad para procesar grandes volúmenes de datos y realizar análisis complejos de manera eficiente. Su integración se ha visto en áreas como la automoción, la financiación, la atención médica y la educación, donde la IA ha revolucionado los enfoques tradicionales para la resolución de problemas y la toma de decisiones. A medida que estas tecnologías maduran, su implementación en sectores especializados promete generar innovaciones disruptivas que podrían redefinir los paradigmas actuales de operación y servicio en muchas industrias.

En cuanto a la medicina, la oftalmología ha sido un campo beneficiado por estas innovaciones, las redes neuronales y la IA han tenido un impacto significativo en áreas de la atención médica, facilitando diagnósticos más precisos y tratamientos más eficaces. Por ejemplo, en la detección temprana de enfermedades como el cáncer, los algoritmos de IA pueden analizar imágenes médicas para identificar anomalías que podrían pasarse por alto en evaluaciones más convencionales [25]. Además, el uso de IA en la telemedicina ha permitido evaluar a pacientes a distancia, mejorando el acceso a cuidados médicos sin necesidad de visitas presenciales, lo que ha sido especialmente crucial durante la pandemia de COVID-19.

En el ámbito de las comunicaciones ópticas, la combinación de óptica adaptativa y redes neuronales se ha convertido en una herramienta indispensable para corregir las distorsiones que afectan a la transmisión de señales a través de la atmósfera. Las fluctuaciones atmosféricas pueden degradar seriamente la calidad de las comunicaciones ópticas, especialmente en largas distancias. La óptica adaptativa inteligente (IAO) utiliza redes neuronales para predecir y corregir estas distorsiones de manera precisa y eficiente en tiempo real. Esto no solo mejora la calidad de las comunicaciones ópticas, sino que también aumenta su fiabilidad, permitiendo una transmisión de datos más rápida y estable. Estas mejoras son cruciales para aplicaciones como la comunicación por satélite y la internet óptica, donde la precisión y la velocidad son esenciales.

En defensa y vigilancia, la óptica adaptativa se ha utilizado para mejorar la calidad de las imágenes obtenidas en condiciones atmosféricas adversas, como niebla, humo o

turbulencia. La IAO ha potenciado estas capacidades al implementar algoritmos de predicción de turbulencias y corrección de aberraciones en tiempo real, lo que mejora significativamente la capacidad de los sistemas de vigilancia para detectar y seguir objetivos. Esto es particularmente importante en escenarios de vigilancia aérea y marítima, donde las condiciones pueden cambiar rápidamente y la capacidad de obtener imágenes claras y precisas puede ser crítica para la seguridad y el éxito de las misiones.

En astronomía, la óptica adaptativa ha sido una herramienta fundamental para corregir las aberraciones causadas por la atmósfera terrestre, permitiendo que los telescopios obtengan imágenes más nítidas y detalladas del universo. La IAO lleva este proceso un paso más allá mediante el uso de algoritmos de aprendizaje profundo que pueden predecir y corregir las aberraciones atmosféricas de manera más efectiva. Esto no solo mejora la calidad de las observaciones astronómicas, sino que también amplía nuestras capacidades de investigación, permitiendo a los astrónomos estudiar objetos celestes con una precisión sin precedentes. La integración de la IAO en telescopios terrestres y espaciales está revolucionando nuestra comprensión del cosmos [14].

Además de estos avances en comunicaciones ópticas, defensa y vigilancia, y astronomía, las redes neuronales y la IA tienen un potencial enorme en otros campos. A medida que los algoritmos de aprendizaje automático y las capacidades de procesamiento continúan avanzando, se espera que la IAO se integre en más sistemas y tecnologías. Estas futuras aplicaciones pueden incluir desde la mejora de los sistemas de imagen médica y la optimización de las redes de comunicación, hasta el desarrollo de nuevas tecnologías en la exploración espacial y la defensa. La capacidad de la IAO para adaptarse y aprender en tiempo real abre un vasto abanico de posibilidades, haciendo que esta tecnología sea una de las más prometedoras y transformadoras del siglo XXI.

En resumen, las redes neuronales y la inteligencia artificial están revolucionando diversos campos tecnológicos. La óptica adaptativa inteligente no solo está mejorando las comunicaciones ópticas y la vigilancia, sino que también está revolucionando la astronomía y tiene un futuro lleno de potencial en una multitud de aplicaciones, haciendo la tecnología más accesible, precisa y eficiente.

4. Herramientas y tecnologías

4.1. HARDWARE UTILIZADO

La ejecución final se ha realizado sobre un servidor bautizado como Hyperion del instituto de ciencias y tecnologías espaciales de Asturias (ICTEA), perteneciente a la Universidad de Oviedo.

Hyperion está diseñado como un nodo de cálculo, lo que significa que su principal función es realizar operaciones computacionales intensivas. Este tipo de nodo es esencial en entornos de investigación y desarrollo donde se ejecutan modelos de aprendizaje automático y redes neuronales profundas. Este equipo no pertenece a ninguna marca comercial específica, ya que ha sido ensamblado y configurado de manera personalizada para satisfacer las necesidades específicas de cómputo. El diseño personalizado permite una mayor flexibilidad y optimización del rendimiento según los requisitos del usuario.

El equipo mantiene la misma denominación para facilitar su identificación y administración dentro de la infraestructura de red y entre otros equipos o nodos que puedan estar en uso.

El procesador *Intel Xeon E5-1650 v3* es un procesador de alto rendimiento perteneciente a la familia de procesadores Intel Xeon, conocida por su robustez y capacidad para manejar cargas de trabajo intensivas. El E5-1650 v3 cuenta con seis núcleos físicos, lo que permite ejecutar múltiples hilos de procesamiento simultáneamente, esencial para tareas de cómputo paralelo en aplicaciones de inteligencia artificial.

El procesador *Intel Xeon E5-1650 v3* ofrece un total de 6 núcleos físicos y, gracias a la tecnología *Hyper-Threading* de Intel, se duplica el número de núcleos virtuales a 12 (6 físicos + 6 virtuales). Esto mejora significativamente la capacidad de procesamiento del equipo, permitiendo una mejor gestión de tareas y mayor eficiencia en la ejecución de aplicaciones multitarea.

El equipo está equipado con 128 GB de memoria RAM. Esta gran cantidad de memoria ayuda en el manejo eficiente de grandes conjuntos de datos para proporcionar un

entorno de ejecución rápido y fluido para aplicaciones de redes neuronales y aprendizaje profundo. La memoria RAM permite almacenar y acceder rápidamente a los datos necesarios durante el entrenamiento de modelos, reduciendo el tiempo total de procesamiento.

El almacenamiento combina velocidad y capacidad. Con un SSD NVMe de 1 TB, el servidor puede acceder y leer datos de forma rápida, lo que ayuda en la ejecución de operaciones de lectura/escrituras intensivas. Además, cuenta con 5 TB en HDD, proporcionando un amplio espacio para el almacenamiento de grandes volúmenes de datos, como conjuntos de datos de entrenamiento, modelos pre-entrenados y resultados de análisis.

El servidor ocupa 4U, lo que indica que ocupa cuatro unidades de altura en un *rack* de servidores estándar. Este formato es común en centros de datos y entornos de servidor debido a su eficiencia en el uso del espacio y la capacidad de gestión térmica. Con una potencia de alimentación total de 1500W, Hyperion está diseñado para manejar componentes de alto rendimiento y garantizar la estabilidad del sistema incluso bajo cargas de trabajo intensivas. Una fuente de alimentación robusta es crucial para mantener el funcionamiento continuo y eficiente del servidor.

Hyperion está equipado con dos GPUs *GTX Titan X*, cada una con 12 GB de VRAM. Estas GPUs son cruciales para el entrenamiento de redes neuronales, proporcionando una capacidad de procesamiento paralelo que acelera significativamente el tiempo de entrenamiento y la eficiencia de los modelos de aprendizaje profundo.

El sistema operativo instalado en Hyperion es Ubuntu 18.04 LTS (*Long Term Support*). Ubuntu es una distribución de Linux ampliamente utilizada en entornos de desarrollo e investigación debido a su estabilidad, seguridad y compatibilidad con una amplia gama de software y bibliotecas de inteligencia artificial y aprendizaje automático.

El equipo fue adquirido en 2015, lo que indica que ha estado en uso durante varios años. Aunque no es de última generación, su configuración robusta aún es adecuada para muchas aplicaciones de inteligencia artificial y procesamiento intensivo de datos.

En resumen, Hyperion es un nodo de cálculo altamente capacitado y personalizado, ideal para el desarrollo y entrenamiento de redes neuronales. Su combinación de un potente

procesador, una amplia memoria RAM, capacidades de almacenamiento considerables y GPUs de alto rendimiento, junto con un sistema operativo estable, lo convierte en una herramienta valiosa para la ejecución de nuestro proyecto.

En la siguiente tabla se enumeran los detalles anteriormente dados:

Nombre del equipo:	Hyperion
Descripción del equipo:	Nodo de cálculo
Marca comercial del	Diseño personalizado
Tipo de procesador:	Intel Xeon E5-1650 v3
Número de núcleos totales:	6 físicos + 6 virtuales
Memoria RAM total:	128 Gb
Capacidad total de almacenamiento:	1Tb SSD + 4Tb HDD
Formato y tamaño:	Rack 4U
Potencia de alimentación:	1500W
Fecha de adquisición:	2015
GPUs disponibles:	2 GTX Titan X con 12Gb de VRAM
Sistema Operativo:	Ubuntu 18.04 LTS

Tabla 1-Información del servidor Hyperion.

4.2. LENGUAJE DE PROGRAMACIÓN

El lenguaje de programación utilizado en este proyecto es Python 3.6.9 Python es un lenguaje de alto nivel, interpretado y de propósito general, conocido por su sintaxis clara y legible, lo que facilita su aprendizaje y uso. Este lenguaje fue creado por Guido van Rossum y lanzado por primera vez en 1991. Python ha ganado popularidad gracias a su

versatilidad y eficiencia, siendo ampliamente utilizado en diversos campos como el desarrollo web, el análisis de datos y la inteligencia artificial.

La elección de Python 3.6.9 en este proyecto se debe a las dependencias específicas del entorno de ejecución en el servidor. Esta versión es ampliamente utilizada y recomendada para nuevos desarrollos debido a su estabilidad y compatibilidad con una gran cantidad de bibliotecas esenciales. El servidor utilizado utiliza Python 2.7.17, pero el entorno en este proyecto requiere Python 3.6.9 para asegurar la correcta integración y funcionamiento de todas las librerías y herramientas necesarias, garantizando así un entorno de desarrollo confiable y eficiente.

4.3. LIBRERÍAS Y ENTORNO DE TRABAJO

Las librerías utilizadas en este proyecto están instaladas en un entorno virtual propio dentro del servidor para evitar conflictos entre dependencias y permitir un desarrollo más organizado. A continuación, se detallan algunas de las librerías más importantes:

- ***NumPy***: Biblioteca fundamental para la computación científica en Python. Proporciona soporte para *arrays* y matrices de grandes dimensiones, junto con una amplia colección de funciones matemáticas de alto nivel para operar con estos *arrays*.
- ***SciPy***: Construida sobre *NumPy*, *SciPy* ofrece funciones adicionales para la computación científica, incluyendo optimización, integración, interpolación, álgebra lineal, estadísticas y más.
- ***Matplotlib***: Biblioteca de trazado 2D que permite la generación de gráficos y visualizaciones de datos en diversos formatos.
- ***TensorFlow***: Plataforma de código abierto para el aprendizaje automático, desarrollada por Google. *TensorFlow* permite la construcción y el entrenamiento de modelos de aprendizaje profundo, y es ampliamente utilizado en tareas de reconocimiento de imágenes, procesamiento de lenguaje natural y más.
- ***Keras***: API de alto nivel para redes neuronales, escrita en Python y capaz de ejecutarse sobre *TensorFlow*. *Keras* facilita la creación rápida y sencilla de prototipos de redes neuronales.

- ***h5py***: Interfaz entre Python y el formato de archivo HDF5, que es ideal para almacenar grandes volúmenes de datos numéricos. *h5py* permite trabajar con estos archivos de manera eficiente, proporcionando una API similar a la de *NumPy*.

Las bibliotecas mencionadas proporcionan un conjunto completo de herramientas para el análisis de datos, visualización, y desarrollo de modelos de aprendizaje automático. Al utilizar un entorno virtual, se asegura que todas las dependencias estén aisladas y gestionadas adecuadamente, permitiendo un desarrollo más eficiente y organizado.

5. Obtención de los datos

La obtención de datos ha sido realizada con la ayuda mis tutores, de manera que podían respaldarme para los experimentos con diferentes *datasets* obtenidos mediante simulación. Estos datos se obtenían modelizando y simulando la turbulencia de la que se había hablado anteriormente en este trabajo para más adelante realizar la propagación de esta mediante una simulación llamada *Split-Step*.

5.1. WOMBAT

WOMBAT es una técnica en el campo de la óptica adaptativa y la física atmosférica diseñada para abordar y corregir las distorsiones causadas por la turbulencia atmosférica en los frentes de onda ópticos, la cual se basa en todos los conceptos teóricos explicados en el apartado 3. Esta metodología se centra en analizar los perfiles de haz de luz mientras atraviesan la atmósfera, lo que permite una medición precisa y una corrección efectiva de las aberraciones del frente de onda. A diferencia de otros métodos de detección de frentes de onda, WOMBAT se distingue principalmente por el tipo de láser que utiliza. Como se explica en la parte teórica, mientras que las estrellas guías láser son puntuales, WOMBAT utiliza dos láseres anchos a través de toda la apertura del telescopio, evitando así el efecto cono. Específicamente, se emplea un láser colimado (recto) y otro desenfocado (que se amplía), lo que permite una reconstrucción más precisa del frente de onda. Un esquema se muestra en la Figura 4.1.

WOMBAT se puede presentar como la evolución de PPPP (*Projected Pupil Plane Pattern*) [26], el cual es una técnica clave en la detección de frentes de onda. El método PPPP implica capturar imágenes de un láser ancho/amplio (*wide laserbeam*), independientemente distantes de la pupila. Estas imágenes se utilizan para determinar las variaciones de intensidad causadas por la difracción y las aberraciones atmosféricas.

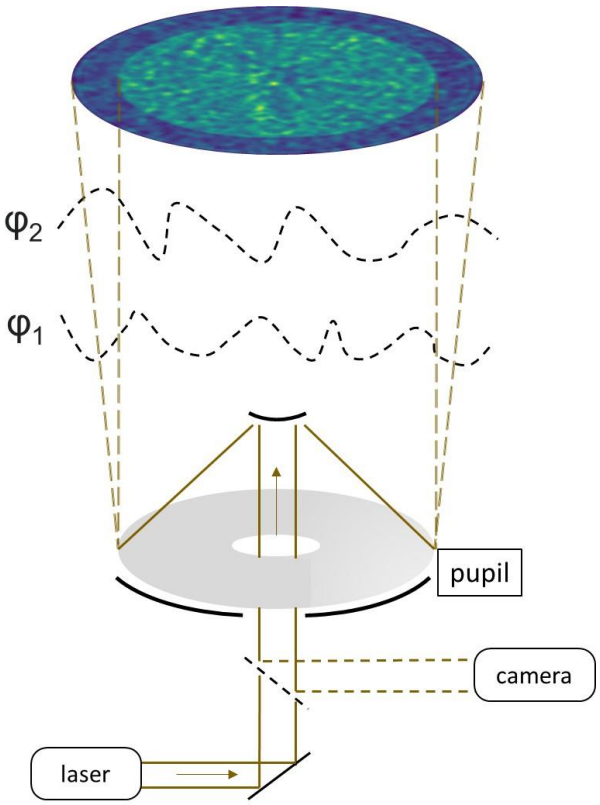

Figura 5.1- Esquema conceptual del método WOMBAT, en el cual dos haces de luz (láser) son emitidos desde un telescopio, uno colimado y otro divergente, que se difractan para producir variaciones de intensidad en los haces. Dicha intensidad emitida se vuelve a recibir en el telescopio y se captura mediante una cámara para producir un perfil de ambos haces. Con dichas imágenes podremos utilizar algoritmos no lineales, como redes neuronales, para obtener información sobre el frente de onda.

La formulación matemática de PPPP emplea la ecuación de transporte de intensidad para relacionar la intensidad y la fase del frente de onda. Al medir la intensidad en dos planos diferentes, es posible aproximar las derivadas del frente de onda y, por ende, reconstruir el frente de onda con cierta precisión. El método PPPP proporciona una base sólida para la medición de frentes de onda, pero presenta limitaciones en cuanto a la precisión y la capacidad de corregir aberraciones más complejas.

WOMBAT se basa en principios similares a los del método PPPP, pero incorpora mejoras significativas. La teoría detrás de WOMBAT se centra en la detección y corrección de las distorsiones en un frente de onda de luz causadas por la turbulencia atmosférica. En el contexto de WOMBAT, se utilizan dos tipos de haz láser: uno es un haz amplio colimado

reflejado en la pupila de un telescopio, y el otro es un haz desenfocado, que por eso es más amplio. El haz colimado permite una mayor cobertura, sobre toda la superficie del telescopio, y precisión en la medición de las distorsiones, mientras que el haz desenfocado ayuda a capturar variaciones más sutiles en la intensidad del haz, proporcionando información adicional crucial para la reconstrucción del frente de onda. Además, mientras en PPPP se tiene una captura en varios planos, una de las mejoras significativa de WOMBAT se basa en la captura en un único plano.

WOMBAT propone un método para prescindir del sensor de frente de onda. En un sistema típico, una parte de la luz se dirigiría al sensor y otra parte a la cámara "de ciencia". Sin embargo, WOMBAT elimina la necesidad de este sensor adicional, ya que solo requiere una imagen del láser en la atmósfera. Este enfoque no solo simplifica el sistema, sino que también resulta más económico, dado que los conjuntos de lentes para sensores de frente de onda son costosos. Así, WOMBAT representa una solución eficiente y asequible para la corrección de aberraciones en los frentes de onda ópticos, mejorando la calidad de las observaciones astronómicas y otros campos que dependen de una óptica adaptativa precisa.

La implementación práctica de WOMBAT requiere solo el uso de una cámara que recoja la imagen del láser en las capas altas de la atmósfera para capturar los perfiles de haz. Estas mediciones se analizan luego para determinar las distorsiones del frente de onda. En un experimento típico, se utiliza un láser pulsado para crear los perfiles de haz, y la intensidad se registra en un plano. Estas mediciones de intensidad se utilizan para reconstruir el frente de onda, corrigiendo las aberraciones introducidas por la turbulencia atmosférica.

La figura 5.1 ilustra cómo WOMBAT utiliza los perfiles de haz capturados en diferentes posiciones para inferir las distorsiones en el frente de onda. Al analizar las variaciones en la intensidad de la luz, es posible calcular y corregir las aberraciones del frente de onda, mejorando la calidad de las observaciones y comunicaciones ópticas.

La combinación de estos dos tipos de haz permite una medición más precisa y detallada de las distorsiones del frente de onda. Además de esto la cámara se utiliza para registrar los perfiles de haz en diferentes planos. Estas imágenes de intensidad se analizan

para reconstruir el frente de onda y corregir las distorsiones causadas por la turbulencia atmosférica.

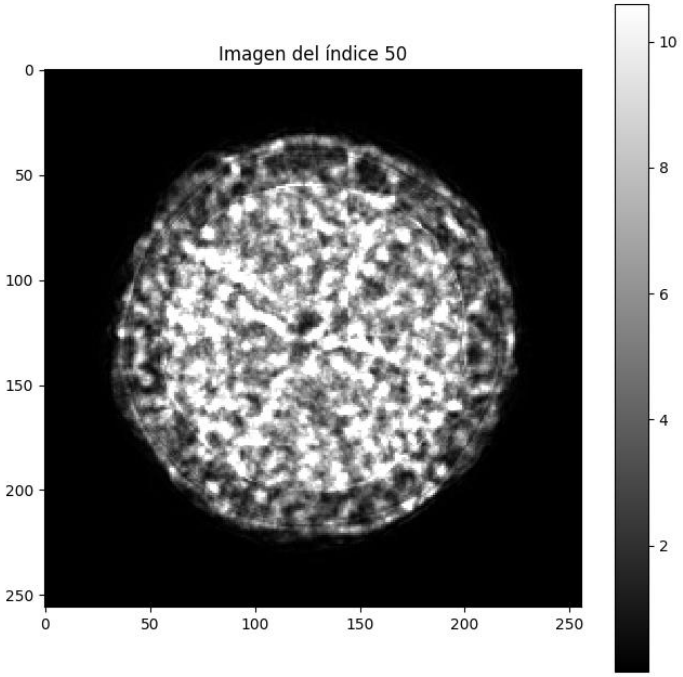

Figura 5.2- Ejemplo de imagen de la propagación de los láseres, obtenida tras una simulación del set de datos.

En la figura 5.2 se puede ver una representación gráfica de la propagación de los láseres mencionados en WOMBAT, apreciándose uno con una apertura menor y otro con una apertura superior.

En la imagen se puede apreciar:

- **Forma circular**: La imagen tiene una forma circular, ya que la propagación de los láseres se ha simulado para que abarque un área circular, similar a cómo se comportarían en un entorno real al propagarse desde la pupila del telescopio. Esto asegura que los datos de propagación sean relevantes para aplicaciones prácticas.
- **Mapa de colores:** La imagen utiliza una escala de grises para representar la intensidad de la propagación de los láseres. Los valores más oscuros indican

una menor intensidad, mientras que los valores más claros representan una mayor intensidad. Este mapa de colores facilita la visualización de las variaciones de intensidad a lo largo de la propagación del haz, además de que será como se introduzcan en nuestra red neuronal.

- **Ejes:** Los ejes X e Y están etiquetados y van desde 0 hasta 250, indicando las coordenadas espaciales sobre las cuales se evalúa la propagación de los láseres. Esto representa el número total de píxeles en ambas direcciones, proporcionando una referencia espacial clara para la interpretación de la imagen.
- **Comparación de aperturas:** En la imagen se pueden distinguir las diferencias en la propagación entre un láser colimado y otro divergente. El análisis de estas diferencias es crucial para entender cómo la apertura del láser afecta la propagación y, por ende, la corrección de aberraciones mediante el método WOMBAT.

Esta imagen es esencial para comprender el comportamiento de los láseres utilizados en WOMBAT y cómo sus características afectan la medición y corrección de frentes de onda en la atmósfera.

Como resultado de la aplicación de WOMBAT también podemos guardar la turbulencia gracias a una serie de polinomios de Zernike.

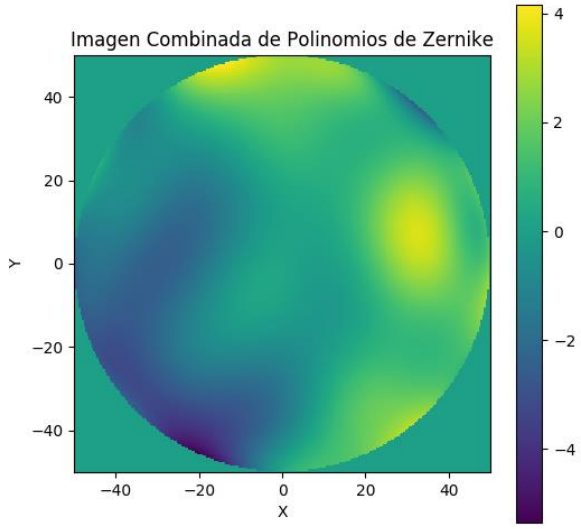

Figura 5.3- Ejemplo de turbulencia reconstruida en base a los primeros 55 polinomios de zernike de un *dataset* simulado.

La imagen que se ha generado de muestra y que se puede observar en la figura 5.3, es una reconstrucción utilizando 55 zernikes. Estos polinomios, como se ha indicado anteriormente, son una serie de funciones ortogonales y que ayudan a describir las aberraciones de las ondas, cuyas principales características son:

- **Forma circular**: La imagen tiene una forma circular puesto los polinomios de Zernike están definidos sobre un círculo unitario. Esto significa que cualquier punto fuera del círculo tiene un valor de cero o no está definido.
- **Mapa de colores**: La imagen utiliza un mapa de colores para representar los valores de la función combinada de polinomios de Zernike. Los colores varían de púrpura (valores negativos) a verde (valores cercanos a cero) y amarillo (valores positivos).
- **Ejes**: Los ejes X e Y están etiquetados y van desde -50 hasta 50, indicando las coordenadas espaciales sobre las cuales se evalúan los polinomios de Zernike y que representan el número total de píxeles de la fase.
- **Combinación de coeficientes**: Esta imagen en particular es el resultado de la combinación de los primeros 55 coeficientes de una fase concreta. Esto significa que se han utilizado los primeros 55 términos de la serie de polinomios de Zernike, con sus respectivos coeficientes, para generar esta visualización.

5.2. SIMULACIÓN DE DATOS ATMOSFÉRICOS

Con las descripciones teóricas del modelo atmosférico presentadas, ahora describimos una manera de simular los efectos que se quieren estudiar.

La modalidad de simulación de *Split-Step* es un enfoque general que se ha utilizado tanto computacionalmente como en un sentido de laboratorio con dispositivos físicos como moduladores de fase sintonizables, fuentes de calor o pantallas de plástico, donde una serie de dispositivos físicos se organizan de tal manera que una onda en propagación es perturbada por el dispositivo.

De manera simple, el método *split-step* es una forma intuitiva de simular los efectos de la turbulencia en la atmósfera. Básicamente, representamos la atmósfera usando pantallas de fase distribuidas a lo largo del camino que recorre la luz. El *split-step* hace lo mismo, pero de forma numérica: divide el trayecto de la luz en varias secciones y simula cómo la onda de luz pasa a través de cada una de esas pantallas.

Los componentes básicos de este modelo de simulación son:

1. **Generación de pantallas de fase**: El primer y más importante paso es crear las pantallas de fase numéricas. Estas pantallas definen la intensidad de la turbulencia y la geometría de la imagen.
2. **Propagación numérica de la onda**: Una vez que se tienen las pantallas de fase, se modela una fuente de luz y se hace pasar a través de estas pantallas. Esto se hace mediante métodos numéricos de propagación de la onda (como los métodos de Fresnel). Es importante destacar que esta propagación y ajuste se realiza para cada píxel de la imagen.

Como se ha mencionado, la propagación numérica de la onda se realiza píxel por píxel, utilizando la propiedad de superposición de ondas. Este proceso es el principal cuello de botella computacional del método *split-step* [27].

La simulación imita el paso de un láser hasta las capas altas de la atmósfera donde se toma una "foto". Esta foto se guarda como una imagen que se utilizará como entrada para las redes neuronales, de la cual se puede escoger el tamaño. Además, los coeficientes

de Zernike, que describen la turbulencia colapsada (una suma ponderada de los efectos de la turbulencia) sobre el telescopio, se guardan como la salida para nuestra red.

Los coeficientes de Zernike son vitales ya que proporcionan una representación compacta de las distorsiones del frente de onda causadas por la turbulencia. Estos coeficientes se derivan ajustando el frente de onda observado a una serie de polinomios de Zernike, que son un conjunto de polinomios ortogonales definidos en un disco unitario. Estos polinomios se utilizan comúnmente en óptica para representar datos del frente de onda, como se indicó anteriormente en el apartado 3.1.1.1.

El método de simulación *Split-Step*, junto con los coeficientes de Zernike, es eficaz para modelar los efectos de la turbulencia atmosférica en los sistemas de imágenes. Extender la simulación a capas altas de la atmósfera y usar pantallas de fase y propagación de ondas permite simular estos efectos con precisión. Los coeficientes de Zernike mejoran la exactitud de la simulación, ayudando a analizar las distorsiones del frente de onda.

Este enfoque es clave para desarrollar técnicas avanzadas de imagen y mejorar el rendimiento de los sistemas ópticos.

6. Metodología de trabajo

En el presente trabajo se utilizarán técnicas de aprendizaje profundo para entrenar un modelo con conjuntos de datos de WOMBAT. Se realizarán pruebas y ajustes iterativos para mejorar la precisión del modelo. La metodología adoptada se detalla a continuación, explicando los pasos específicos y las diferentes decisiones tomadas durante el desarrollo del proyecto.

6.1. SELECCIÓN DEL MODELO BASE

El modelo inicial utilizado en este proyecto es una red neuronal convolucional diseñada para procesar y analizar los datos de tomografías obtenidos, inspirado en experimentos previos [28]. La arquitectura del modelo incluye múltiples capas convolucionales seguidas de capas de *pooling* y finalmente capas totalmente conectadas. Las funciones de activación utilizadas incluyen *ReLU* para las capas convolucionales y una función sigmoide para la capa de salida.

Este modelo se ha obtenido a través de diferentes pruebas iniciales y modificaciones de conjunto de datos más pequeños, partiendo de la base citada anteriormente, trabajando inicialmente en la herramienta Google Colab, hasta llegar al punto de transición a sets de datos de gran tamaño y trabajo en el servidor Hyperion.

La arquitectura del modelo inicial será por lo tanto la siguiente:

- Capas convolucionales:
  - Primera capa convolucional:
    - Filtros: 32
    - Tamaño del Kernel: 3
    - Activación: ReLU
    - BatchNormalization para estabilizar y acelerar el proceso de entrenamiento.

        - MaxPooling2D para reducir la dimensionalidad espacial (2x2).

    - Segunda capa convolucional:
        - Filtros: 64
        - Tamaño del Kernel: 3
        - Activación: ReLU
        - BatchNormalization.
        - MaxPooling2D (2x2).

    - Tercera capa convolucional:
        - Filtros: 128
        - Tamaño del Kernel: 3
        - Activación: ReLU
        - BatchNormalization.
        - MaxPooling2D (2x2).

- Capas de Pooling y regularización:
    - ***GlobalAveragePooling2D***: Después de las capas convolucionales y de pooling para reducir aún más la dimensionalidad y preparar la entrada para las capas totalmente conectadas.
    - ***Dropout***: 30% después de la capa de *GlobalAveragePooling2D* para evitar el sobreajuste.

- Capa de salida:
    - Una capa densa de salida con 55 unidades, correspondiente al número de coeficientes a predecir en la reconstrucción tomográfica.

El optimizador utilizado es Adam, que es una combinación de dos de los optimizadores más populares: AdaGrad, que ajusta la tasa de aprendizaje para cada parámetro basándose en los gradientes acumulados, y RMSProp, que utiliza una media móvil exponencial de los gradientes al cuadrado para mantener la tasa de aprendizaje estable. Adam se emplea con una tasa de aprendizaje de inicio de 0.0001.

En cuanto a la función de coste utilizada se emplea el error cuadrático medio (MSE), que además permite evaluar la precisión del modelo durante el entrenamiento y la validación.

Todo el modelo base puede verse de forma abreviada en la tabla 2, la cual muestra un resumen de este:

Capa	Descripción
Convolucional 1	Filtros: 32, Tamaño de Kernel: 3, Activación: ReLU, BatchNormalization, MaxPooling2D(2x2)
Convolucional 2	Filtros: 64, Tamaño de Kernel: 3, Activación: ReLU, BatchNormalization, MaxPooling2D(2x2)
Convolucional 3	Filtros: 128, Tamaño de Kernel: 3, Activación: ReLU, BatchNormalization, MaxPooling2D(2x2)
GlobalAveragePooling2D	Reduce cada mapa de características a un único valor promedio
Dropout	Tasa de Dropout: 30%
Capa Densa de Salida	Unidades: 55, Activación: Lineal
Optimizador	Adam, Tasa de Aprendizaje: 0.0001
Métrica de Evaluación	Mean Square Error (MSE)

Tabla 2-Tabla resumen del modelo base de la red a entrenar.

También se han implementado diferentes funciones auxiliares (denominadas *callbacks* en TensorFlow) para aumentar la eficiencia del entrenamiento:

- **EarlyStopping**: Implementado para detener el entrenamiento cuando la mejora en la validación se estanca. Este *callback* monitorea la pérdida de validación y si no hay mejora después de un número determinado de épocas, detiene el entrenamiento para evitar el sobreajuste y ahorrar recursos computacionales.
- **ReduceLROnPlateau**: Utilizado para reducir la tasa de aprendizaje cuando el rendimiento en el conjunto de validación deja de mejorar. Si la pérdida de validación no mejora después de un número específico de épocas, este *callback* reduce la tasa de aprendizaje en un factor determinado, ayudando al modelo a converger de manera más eficiente.

- **ModelCheckpoint**: Para guardar el modelo con el mejor rendimiento durante el entrenamiento. Este *callback* guarda el estado del modelo cada vez que se obtiene un nuevo mejor resultado en términos de pérdida de validación, permitiendo recuperar el mejor modelo alcanzado durante el entrenamiento.

Por último, también se han implementado normalización de los datos utilizados, de manera que la dificultad de aprendizaje del modelo disminuyese:

- **Imágenes**: Las imágenes se normalizan en el rango [0, 1]. Esta normalización permite estandarizar los datos de entrada, mejorar la estabilidad y la eficiencia del entrenamiento del modelo.
- **Coeficientes**: Los coeficientes se normalizan utilizando el método *scaler.fit_transform* para el conjunto de entrenamiento y *scaler.transform*, utilizadas a partir del objeto *powertransformer* proveniente de la biblioteca *Scikit-learn* para el conjunto de validación y prueba, asegurando que todos los datos sigan la misma escala de normalización, aproximándose a una distribución normal (gaussiana). Esto ayuda a que el modelo aprenda de manera consistente y que los coeficientes de salida estén en un rango adecuado para la tarea de reconstrucción tomográfica.

Esta configuración de la red y el proceso de normalización aseguran que el modelo pueda aprender de manera eficiente y generalizar bien a datos no vistos, maximizando su rendimiento en la tarea de reconstrucción tomográfica a través del entrenamiento. Además, el uso de *callbacks* y técnicas de normalización de datos permite que el modelo no solo sea preciso sino también robusto y eficiente en términos de recursos computacionales.

6.2. DATASETS DE TRABAJO

En total se utilizan 3 set de datos, los cuales se combinan en un solo conjunto, para entrenamiento y 3 *datasets* diferentes para validación. Cada set de datos se compone de 50 mil imágenes de 255px x 255px, cada imagen con sus correspondientes 55 coeficientes.

Para el entrenamiento se utilizará train_2atm. Este *dataset* se genera combinando y seleccionando datos de varios archivos de *datasets* con dos capas atmosféricas, pero con diferentes valores para r_0. Con esto se consigue un conjunto de datos "simple", pues solo tiene dos capas, pero con cierta complejidad al combinar distintos valores de r_0. Los sets de datos de origen son 008_2atm, 012_2atm_1 y 016_2atm_1.

En cuanto a la evaluación se realizará con estos tres *datasets*, que contienen datos diferentes a los utilizados para el entrenamiento, aunque posee algunos parámetros de simulación similares:

- Dataset: 012_2atm:
  - 012: Representando el r_0 (parámetro de coherencia).
  - 2atm: Número de capas atmosféricas simuladas. En este caso con igual complejidad al de entrenamiento y por lo tanto el que mejores resultados nos generará.
  - Propósito: Este dataset se utiliza para evaluar el rendimiento del modelo en condiciones de dos capas atmosféricas. Se trata del dataset de evaluación principal que establece la referencia para el rendimiento del modelo en condiciones menos complejas.

- Dataset: 012_5atm:
  - 012: Identificador del conjunto de datos, representando el r_0.
  - 5atm: Número de capas atmosféricas simuladas. Con una complejidad superior al de 2 capas, pero aún con un posible buen funcionamiento en la red.
  - Propósito: Utilizado para evaluar la capacidad del modelo para generalizar en condiciones más complejas con cinco capas atmosféricas simuladas. Este *dataset* permite medir cómo el modelo maneja un aumento en la complejidad de las condiciones atmosféricas.

- Dataset: 012_10atm:
  - 012: Identificador del conjunto de datos, representando el r_0.
  - 10atm: Número de capas atmosféricas simuladas. Este es el caso por evaluar más complejo y con el que teóricamente obtendremos peores resultados.
  - Propósito: Evaluar el rendimiento del modelo en condiciones extremadamente complejas con diez capas atmosféricas simuladas. Este *dataset* se utiliza para probar los límites de generalización del modelo bajo condiciones atmosféricas muy adversas.

7. Experimentos y resultados

En este apartado, se presentan los diferentes experimentos realizados junto con los resultados obtenidos y una discusión sobre cuáles fueron las configuraciones más efectivas. En los casos considerados relevantes, se incluirán las gráficas de aprendizaje que ilustran el comportamiento del modelo durante el entrenamiento y la validación.

Es importante destacar que las pruebas se llevarán a cabo de forma secuencial, es decir, cada prueba se realizará sobre la base de la anterior. Si una prueba mejora los resultados, la siguiente tomará como base esta configuración; en caso contrario, se obviará y se continuará con la configuración anterior. Comenzando por la configuración explicada en el apartado 6.

Debido a la complejidad del problema y al elevado número de posibles configuraciones, la cantidad de pruebas potenciales es altísima y escala de forma exponencial. Además, el tiempo de entrenamiento de cada modelo varía entre 90 minutos y 300 minutos. Por estas razones, se ha tenido que acotar bastante el número de opciones posibles realizando una serie de pruebas discutidas con los tutores del proyecto, seleccionando aquellas que se consideraron más pertinentes y prometedoras.

Las pruebas se han realizado efectuando cambios en el modelo base que se presenta en el siguiente apartado 6.1, el cual es entrenado con el *dataset* nombrado train_2atm y este compuesto por los ya nombrados conjuntos de datos.

Seguidamente se han evaluado los modelos ya entrenados con dicho set de datos. Para ello se ha tratado de predecir los coeficientes de los 3 *datasets* de evaluación, realizando evaluaciones individuales, para así compararlos con los reales y poder obtener el error de frente de onda sin corregir (intrínseco o que el *dataset* conlleva por defecto), el error de frente de onda residual y el porcentaje de mejora.

Como se ha nombrado anteriormente, el conjunto más similar al set de entrenamiento, es decir el de 2 capas atmosféricas, será el que mejores resultados obtendrá, y el de 10 capas será el que peores resultados obtendrá. Pese a esto, evaluar el modelo entrenado con un conjunto de datos simple en sets complejos ayudará a ver los límites a los que llegará la red y como se puede acomodar a diferentes casos.

7.1. OBTENCIÓN DE RESULTADOS CON EL MODELO BASE

El modelo base se entrenó inicialmente para establecer un punto de referencia sobre el cual comparar las futuras modificaciones. Los resultados obtenidos con el modelo base se detallan a continuación.

El modelo base se puede consultar en el apartado 6.1 y más concretamente en la tabla 2, la cual lo resume.

Dataset	Error de frente de onda residual (nm)	Error de frente de onda sin corregir (nm)	Porcentaje de mejora del modelo
012_2atm	75	169	55.79%
012_5atm	93	169	45.35%
012_10atm	110	169	34.88%

Tabla 3-Muestra de los diferentes resultados obtenidos tras evaluar el modelo base con los 3 datasets de evaluación.

Como se observa en la tabla 3, los resultados obtenidos con el modelo base muestran una mejora significativa respecto al error sin corregir del set de datos original. Este error muestra el error promedio que tienen todas las imágenes de cada uno de los sets de datos de validación y cuyo valor será el mismo en todos los experimentos. El modelo logra un porcentaje de mejora del 55.79% para el caso con 2 atmósferas, 45.35% para el escenario con 5 atmósferas y 34.88% para el dataset con 10 atmósferas. Este porcentaje de mejora se obtiene al comparar los coeficientes predichos por nuestro modelo con los coeficientes que recrean la turbulencia, según el error presente en el conjunto de datos, proporcionando un error de frente de onda residual, que es el error que la red neuronal no ha sido capaz de corregir.

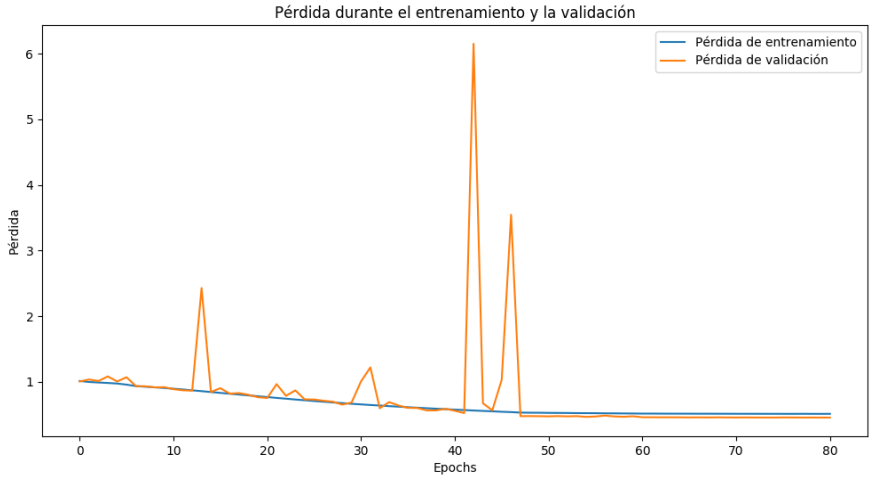

Figura 7.1- Pérdida durante el entrenamiento y la validación en el modelo base.

Como se indica en la figura 7.1, en las primeras 10 épocas, tanto la pérdida de entrenamiento como la de validación disminuyen de manera constante, lo que sugiere que el modelo está aprendiendo adecuadamente y reduciendo el error. Entre las épocas 10 y 50, se observan varios picos en la pérdida de validación, que pueden ser indicativos de inestabilidad en el proceso de entrenamiento. Estos picos pueden ser causados por varias razones, como fluctuaciones en el gradiente, sobreajuste temporal o variabilidad inherente en los datos de validación. Un pico particularmente alto en la pérdida de validación se observa alrededor de la época 45.

Después de la época 50, tanto la pérdida de entrenamiento como la de validación se estabilizan y convergen hacia un valor bajo. Esta estabilización sugiere que el modelo ha alcanzado un punto de equilibrio donde las actualizaciones de los parámetros tienen un impacto menor en la reducción del error.

7.2. ESTUDIO DE VARIACIONES SOBRE EL CASO BASE

7.2.1. LeakyReLU

En esta primera prueba se ha cambiado la función de activación en las capas convolucionales de ReLU a LeakyReLU. Puesto que permite un pequeño gradiente cuando

la unidad no está activa, lo que puede ayudar a mitigar el problema del desvanecimiento de las neuronas que ocurre en ReLU.

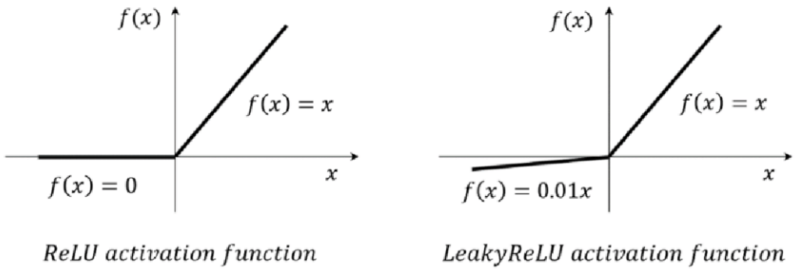

Figura 7.2- Imagen ilustrativa de cómo son ambas funciones de activación, tanto Relu como LeakyReLU.

Como podemos ver en la figura 7.2, la función de activación ReLU (*Rectified Linear Unit*) es ampliamente utilizada en redes neuronales por su simplicidad y efectividad. Sin embargo, puede causar el problema de "neuronas muertas", donde algunas neuronas dejan de aprender si quedan atrapadas en la región cero.

Para mitigar este problema, usamos la función LeakyReLU. Esta función permite pequeños valores negativos, asegurando que las neuronas no queden completamente inactivas y mejorando así la convergencia y rendimiento del modelo.

Aplicamos LeakyReLU en la primera prueba para abordar el problema de las "neuronas muertas" y evaluar si esta modificación mejora el rendimiento del modelo al mantener una pequeña gradiente incluso para entradas negativas.

Dataset	Error de frente de onda residual (nm)	Error de frente de onda sin corregir (nm)	Porcentaje de mejora del modelo
012_2atm	99	169	41.23%
012_5atm	114	169	32.72%
012_10atm	127	169	24.62%

Tabla 4-Resultados obtenidos al modificar la red con LeakyReLU.

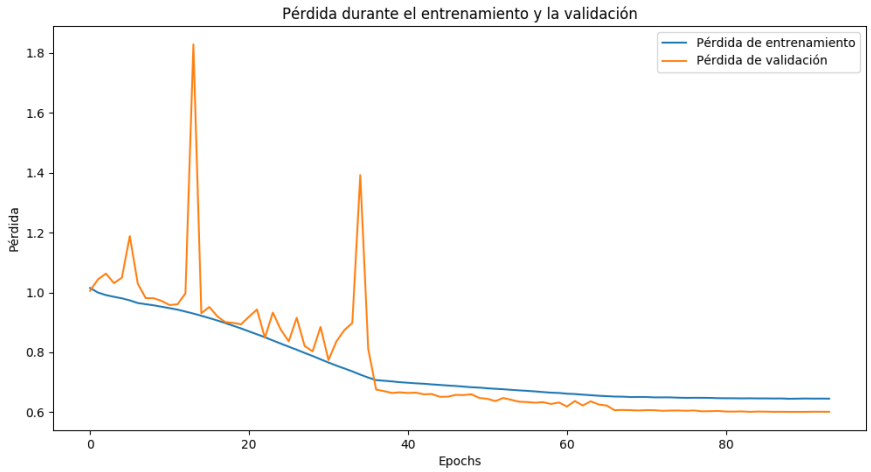

Figura 7.3- Pérdida durante el entrenamiento y la validación en la primera prueba.

En la gráfica de la Figura 7.3 se observa la evolución de la pérdida durante el entrenamiento y la validación del modelo con LeakyReLU a lo largo de 80 épocas.

Durante las primeras épocas, tanto la pérdida de entrenamiento como la de validación disminuyen constantemente, lo que indica que el modelo está aprendiendo. Sin embargo, la tasa de disminución es más lenta en comparación con el modelo base. Además, se observan fluctuaciones y picos en la pérdida de validación que son más pronunciados con LeakyReLU. Hay un pico significativo alrededor de la época 40, que es más alto y sostenido que los observados en el modelo base. Esto sugiere que el uso de LeakyReLU introduce una mayor inestabilidad durante el entrenamiento.

Después de la época 60, tanto la pérdida de entrenamiento como la de validación se estabilizan. Sin embargo, los valores finales de la pérdida de validación son mayores en comparación con el modelo base. La cercanía entre las pérdidas de entrenamiento y validación en las últimas épocas indica que el modelo no está sobreajustando significativamente, pero la capacidad de generalización parece ser inferior.

Aunque el modelo con LeakyReLU mejora respecto al error no corregido del set de datos, el porcentaje de mejora es menor en comparación con el modelo base. Como vemos en la tabla 4 para el *dataset* con 2 atmósferas, la mejora es del 41.23% frente al 55.79% del modelo base. De forma similar, para los *datasets* con 5 y 10 atmósferas, los porcentajes de mejora son del 32.72% y 24.62%, respectivamente, en comparación con el 45.35% y 34.88% del modelo base.

La mayor inestabilidad observada durante el entrenamiento y la validación sugiere que LeakyReLU puede no ser la función de activación más adecuada para este tipo de datos y arquitectura de modelo. Los picos y fluctuaciones pueden indicar que el modelo está teniendo dificultades para aprender características consistentes cuando se utilizan capas LeakyReLU.

Dado que los resultados obtenidos con LeakyReLU son peores en comparación con los obtenidos con ReLU, este cambio será descartado en futuras pruebas. El modelo base con ReLU muestra una mejor capacidad de generalización y menor pérdida tanto en el conjunto de entrenamiento como en el de validación.

7.2.2. Normalización "Yeo-Johnson"

En esta segunda prueba, se implementó la transformación de Yeo-Johnson. La transformación de Yeo-Johnson es una técnica utilizada para normalizar los datos, haciéndolos más cercanos a una distribución normal, lo que puede mejorar el rendimiento de los modelos de aprendizaje automático al estabilizar la varianza y hacer que los datos sean más compatibles con las suposiciones de los algoritmos [29].

Yeo-Johnson se basa en las ecuaciones:

$$y(\lambda) = \begin{cases} \frac{((y+1)^{\lambda}-1)}{\lambda} & si\ \lambda \neq 0 \\ \log(y+1) & si\ \lambda = 0 \end{cases} \quad (14)$$

Para $y \geq 0$, y

$$y(\lambda) = \begin{cases} \frac{((1-y)^{2-\lambda}-1)}{2-\lambda} & si\ \lambda \neq 2 \\ -\log(1-y) & si\ \lambda = 2 \end{cases} \quad (15)$$

Para $y < 0$.

Estas fórmulas transforman los datos dependiendo del valor de λ, ajustándose para mantener la estabilidad de la varianza y mejorar la normalidad de los datos.

Dataset	Error de frente de onda residual (nm)	Error de frente de onda sin corregir (nm)	Porcentaje de mejora del modelo
012_2atm	66	169	61.22%
012_5atm	88	169	48.16%
012_10atm	108	169	35.97%

Tabla 5-Resultados obtenidos tras evaluar el modelo en la aplicando la normalización Yeo-Johnson.

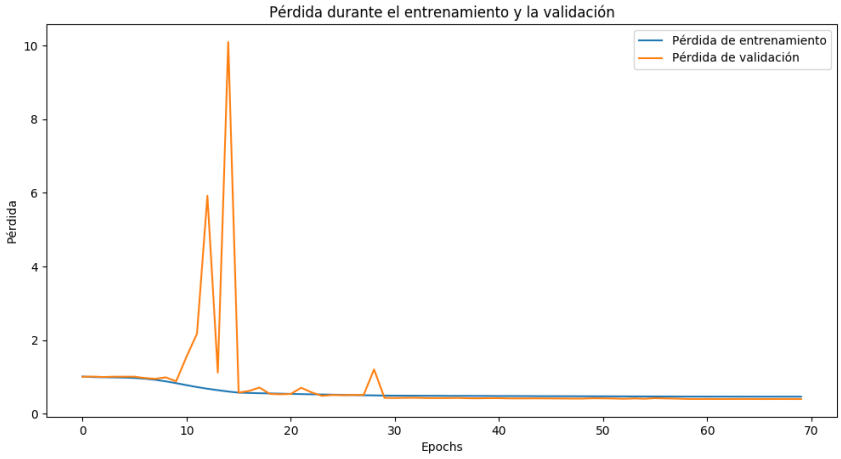

Figura 7.4- Pérdida durante el entrenamiento y la validación en la segunda prueba.

En la Figura 7.4 se puede ver cómo durante las primeras 10 épocas, tanto la pérdida de entrenamiento como la de validación disminuyen de manera constante, lo que indica que el modelo está aprendiendo. Sin embargo, alrededor de la época 20, se observa un pico significativo en la pérdida de validación, seguido por varios picos menores hasta aproximadamente la época 30. Este comportamiento sugiere que el modelo experimenta cierta inestabilidad durante estas épocas.

Después de la época 30, la pérdida de validación disminuye y se estabiliza, convergiendo hacia un valor bajo similar al de la pérdida de entrenamiento. Esto sugiere que, a pesar de las fluctuaciones iniciales, el modelo con la transformación de Yeo-Johnson logra estabilizarse y mejorar su rendimiento.

Los resultados que podemos ver en la tabla 5 muestran una mejora notable en comparación con el modelo base. Para el *dataset* con 2 atmósferas, la pérdida en el conjunto de prueba en nanómetros es de 66, lo que representa un porcentaje de mejora del 61.22%

en comparación con el error intrínseco del *dataset*. Similarmente, para los *datasets* con 5 y 10 atmósferas, los porcentajes de mejora son del 48.16% y 35.97%, respectivamente.

La transformación de Yeo-Johnson parece haber ayudado a normalizar los datos, lo que se refleja en la mejora del rendimiento del modelo. A pesar de las fluctuaciones iniciales en la pérdida de validación, el modelo logra estabilizarse y generalizar bien a los datos de prueba.

7.2.3. Aumento del tamaño del filtro de la primera capa convolucional a 5

En esta tercera prueba, se modificó el tamaño del *kernel* en las capas convolucionales de 3 a 5. Aumentar el tamaño del *kernel* puede ayudar al modelo a capturar características más amplias en las imágenes, lo cual puede mejorar el rendimiento en algunos casos.

Dataset	Error de frente de onda residual (nm)	Error de frente de onda sin corregir (nm)	Porcentaje de mejora del modelo
012_2atm	50	169	70.42%
012_5atm	75	169	55.59%
012_10atm	98	169	41.89%

Tabla 6-Resultados obtenidos tras evaluar el modelo cambiando el tamaño del filtro de la primera capa convolucional.

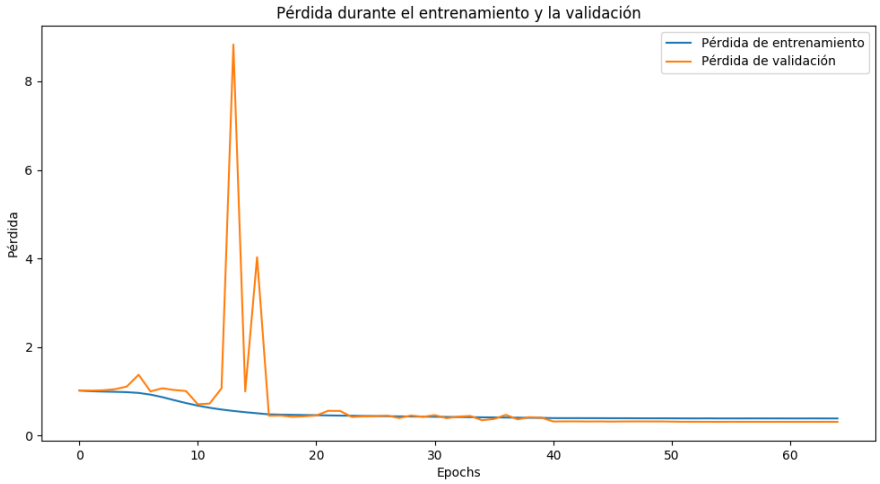

Figura 7.5- Pérdida durante el entrenamiento y la validación en la tercera prueba.

Durante las primeras épocas, tanto la pérdida de entrenamiento como la de validación disminuyen de manera constante, lo que indica que el modelo está aprendiendo adecuadamente. Sin embargo, alrededor de la época 15, se observa un pico significativo en la pérdida de validación, seguido por algunos picos menores hasta aproximadamente la época 25. Este comportamiento sugiere que el modelo experimenta cierta inestabilidad durante estas épocas.

Después de la época 25, la pérdida de validación disminuye y se estabiliza, convergiendo hacia un valor bajo similar al de la pérdida de entrenamiento. Esto sugiere que, a pesar de las fluctuaciones iniciales, el modelo con el tamaño de *kernel* de 5 logra estabilizarse y mejorar su rendimiento.

Los resultados que observamos en la tabla 6 muestran una mejora notable en comparación con el modelo base. Para el *dataset* con 2 atmósferas, la pérdida en el conjunto de prueba en nanómetros es de 50, lo que representa un porcentaje de mejora del 70.42% en comparación con el error intrínseco del *dataset*. Similarmente, para los *datasets* con 5 y 10 atmósferas, los porcentajes de mejora son del 55.59% y 41.89%, respectivamente.

El aumento del tamaño del filtro parece haber ayudado a capturar características más amplias en las imágenes, lo que se refleja en la mejora del rendimiento del modelo. A pesar de las fluctuaciones iniciales en la pérdida de validación, el modelo logra estabilizarse y generalizar bien a los datos de prueba.

Dado que el cambio en el tamaño del filtro a 5 ha mostrado mejoras significativas en comparación con el modelo base, se considerará retener este cambio en futuras pruebas. La estabilización final y la mejora en los porcentajes de mejora del modelo sugieren que esta modificación es beneficiosa para el rendimiento del modelo.

7.2.4. Aumento del tamaño de filtro de la segunda capa convolucional a 4

En esta cuarta prueba, se modificó el tamaño del *kernel* de la segunda capa convolucional de 3 a 4. El objetivo de este cambio es observar si un tamaño de *kernel* ligeramente mayor en una capa intermedia puede capturar mejor las características espaciales de las imágenes y, por ende, mejorar el rendimiento del modelo. Anteriormente, se había probado aumentar el tamaño del *kernel* de 3 a 5, pero en esta ocasión, se opta por un incremento más moderado para evitar aumentar en exceso el número de pesos del modelo. Este enfoque busca un balance entre la capacidad de capturar características más complejas y la eficiencia del modelo, manteniendo un número de parámetros manejable para evitar sobreajuste y mantener un entrenamiento eficiente.

Dataset	Error de frente de onda residual (nm)	Error de frente de onda sin corregir (nm)	Porcentaje de mejora del modelo
012_2atm	32	169	61.22%
012_5atm	64	169	62.42%
012_10atm	91	169	46.14%

Tabla 7-Resultados obtenidos tras aumentar el tamaño del filtro de la segunda capa convolucional.

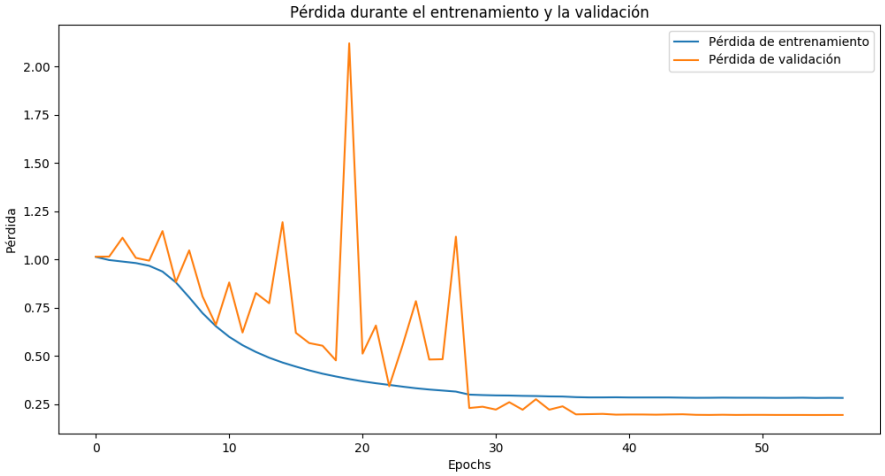

Figura 7.6- Pérdida durante el entrenamiento y la validación en la cuarta prueba.

Al analizar la gráfica, se observa que durante las primeras 10 épocas, la pérdida de entrenamiento disminuye mientras que la pérdida de validación presenta fluctuaciones notables. Estas fluctuaciones indican una inconsistencia en el rendimiento del modelo durante el proceso de ajuste. Sin embargo, a partir de la época 30, la pérdida de validación disminuye y se estabiliza, convergiendo hacia un valor bajo similar al de la pérdida de entrenamiento.

Los resultados obtenidos y que se muestran en la tabla 7 permiten ver una mejora significativa en comparación con el modelo base. Para el *dataset* con 10 atmósferas, la pérdida en el conjunto de prueba en nanómetros es de 91, lo que representa un porcentaje de mejora del 46.14% en comparación con el error intrínseco del *dataset*. Similarmente, para los *datasets* con 5 y 2 atmósferas, los porcentajes de mejora son del 62.42% y 81.29%, respectivamente.

El aumento del tamaño del *kernel* en la segunda capa convolucional parece haber ayudado a capturar características más específicas de las imágenes, lo que se refleja en la mejora del rendimiento del modelo. A pesar de las fluctuaciones iniciales en la pérdida de validación, el modelo logra estabilizarse y generalizar bien a los datos de prueba.

Dado que el aumento del tamaño del *kernel* a 4 en la segunda capa convolucional ha mostrado mejoras significativas en comparación con el modelo base, cogeremos este cambio en futuras pruebas. La estabilización final y la mejora en los porcentajes de mejora del modelo sugieren que esta modificación es beneficiosa para el rendimiento del modelo.

7.2.5. Aumento del *dropout* al 40%

En esta quinta prueba, se implementó el aumento del *dropout* al 40%. El objetivo de aumentar el *dropout* es prevenir el sobreajuste haciendo que el modelo sea más robusto al introducir una mayor aleatoriedad durante el entrenamiento.

Dataset	Error de frente de onda residual (nm)	Error de frente de onda sin corregir (nm)	Porcentaje de mejora del modelo
012_2atm	74	169	56.18%
012_5atm	94	169	44.51%
012_10atm	111	169	33.82%

Tabla 8-Diferentes resultados tras aumentar el dropout al 40%.

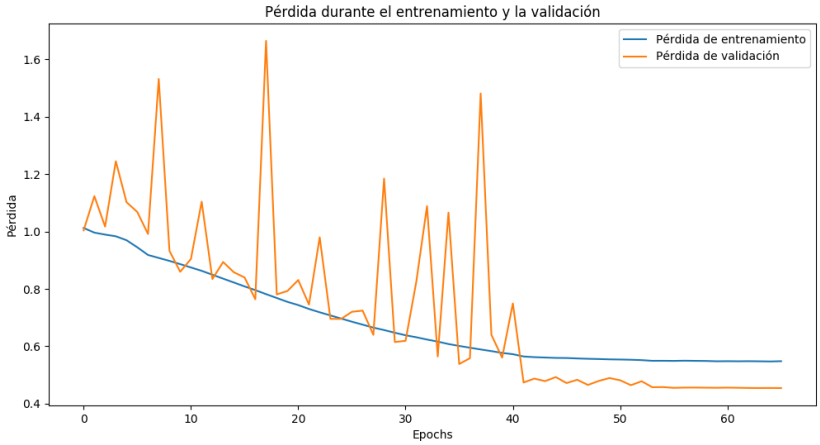

Figura 7.7- Pérdida durante el entrenamiento y la validación en la quinta prueba.

Al analizar la gráfica, se observa que durante las primeras 10 épocas, tanto la pérdida de entrenamiento como la de validación presentan una disminución constante. Sin embargo, a partir de este punto, la pérdida de validación muestra fluctuaciones significativas con picos notables alrededor de las épocas 10, 20, 30 y 40. Estas fluctuaciones pueden indicar que el modelo experimenta inestabilidad en la capacidad de generalización durante el entrenamiento.

A pesar de estas fluctuaciones, se observa que la pérdida de validación tiende a disminuir y estabilizarse a partir de la época 50. La pérdida de entrenamiento sigue una tendencia descendente más suave pero constante, lo que sugiere que el modelo está

aprendiendo de manera efectiva, aunque con cierta variabilidad en el rendimiento de validación.

En la tabla 8 podemos ver que los resultados muestran mejoras en comparación con el modelo base, pero no superan a los obtenidos en la prueba anterior donde se aumentó el tamaño del *kernel* de la segunda capa convolucional a 4. Para el *dataset* con 2 atmósferas, la pérdida en el conjunto de prueba en nanómetros es de 74, con un porcentaje de mejora del 56.18%. Para los *datasets* con 5 y 10 atmósferas, los porcentajes de mejora son del 44.51% y 33.83%, respectivamente.

Aunque el aumento del *dropout* parece haber ayudado a prevenir el sobreajuste, este cambio no resulta en una mejora significativa respecto a la configuración con el *kernel* más grande.

Dado que el cambio realizado no ha mostrado una mejora significativa en comparación con la prueba anterior, este cambio no será retenido en futuras pruebas. El enfoque con el tamaño de *kernel* mayor en la segunda capa convolucional sigue siendo el más prometedor hasta ahora.

7.2.6. Capa totalmente conectada adicional

En esta sexta prueba, se añadió una capa densa adicional al modelo con 128 neuronas. La inclusión de capas densas adicionales puede permitir al modelo capturar interacciones más complejas entre las características extraídas por las capas convolucionales, mejorando así su capacidad de predicción.

Dataset	Error de frente de onda residual (nm)	Error de frente de onda sin corregir (nm)	Porcentaje de mejora del modelo
012_2atm	40	169	76.61%
012_5atm	71	169	58.03%
012_10atm	96	169	42.92%

Tabla 9-Resultados obtenidos tras añadir una capa densa más y evaluar el modelo.

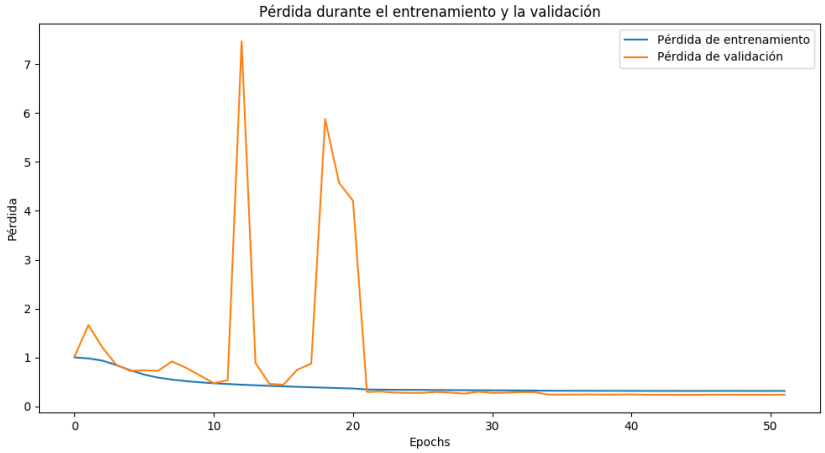

Figura 7.8- Pérdida durante el entrenamiento y la validación en la sexta prueba.

La gráfica muestra una disminución inicial en la pérdida tanto de entrenamiento como de validación durante las primeras 10 épocas. Sin embargo, hay fluctuaciones significativas en la pérdida de validación, con picos notables alrededor de las épocas 10 y 20. A partir de la época 30, la pérdida de validación se estabiliza y converge hacia un valor bajo similar al de la pérdida de entrenamiento.

Los resultados muestran una mejora notable en comparación con el modelo base y la mayoría de las pruebas anteriores. Para el *dataset* con 2 atmósferas, la pérdida en el conjunto de prueba en nanómetros es de 40, lo que representa un porcentaje de mejora del 76.61% en comparación con el error intrínseco del *dataset*. Similarmente, para los *datasets* con 5 y 10 atmósferas, los porcentajes de mejora son del 58.03% y 42.93%, respectivamente.

La inclusión de una capa densa adicional parece haber mejorado la capacidad del modelo para capturar interacciones más complejas entre las características extraídas, lo que se refleja en la mejora del rendimiento del modelo. Aunque hay fluctuaciones iniciales en la pérdida de validación, el modelo logra estabilizarse y generalizar bien a los datos de prueba.

Aunque la adición de una capa densa adicional ha mostrado mejoras significativas en comparación con el modelo base, la prueba 4, que implicó aumentar el tamaño del *kernel* de la segunda capa convolucional a 4, obtuvo mejores resultados globales. Por lo tanto, no se seguirán con los cambios introducidos en esta sexta prueba. La configuración de la prueba 4 seguirá siendo la más prometedora hasta el momento.

7.2.7. Eliminación de una capa convolucional

En esta séptima prueba, se eliminó una capa convolucional del modelo original. La eliminación de capas convolucionales puede reducir la complejidad del modelo y potencialmente mejorar la generalización al evitar el sobreajuste, aunque también puede limitar la capacidad del modelo para capturar características complejas.

En concreto se eliminó la última capa convolucional del modelo. La justificación para esta eliminación es que, al reducir la profundidad del modelo, podemos observar si las capas iniciales son suficientes para capturar las características esenciales de los datos, mientras que la eliminación de la última capa podría simplificar el modelo y potencialmente reducir el riesgo de sobreajuste. Además, la última capa convolucional suele ser responsable de refinar las características extraídas por las capas anteriores; al eliminarla, estamos evaluando si esta refinación adicional es necesaria para mantener un buen rendimiento del modelo.

Dataset	Error de frente de onda residual (nm)	Error de frente de onda sin corregir (nm)	Porcentaje de mejora del modelo
012_2atm	166	169	1.73%
012_5atm	167	169	1.36%
012_10atm	167	169	1.02%

Tabla 10-Resultados obtenidos tras evaluar el modelo con una capa convolucional menos.

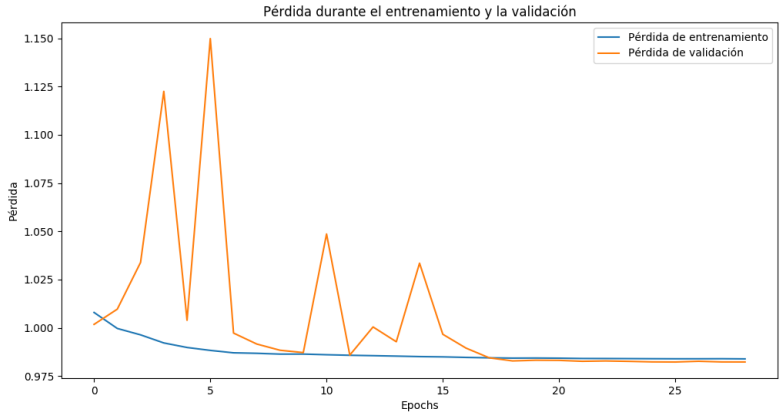

Figura 7.9- Pérdida durante el entrenamiento y la validación en la séptima prueba.

La gráfica muestra una disminución inicial en la pérdida tanto de entrenamiento como de validación durante las primeras épocas. Sin embargo, se observan fluctuaciones significativas en la pérdida de validación, con picos notables alrededor de las épocas 5 y 10. La pérdida de validación se estabiliza en torno a la época 20, convergiendo hacia un valor que es cercano, pero no inferior al de la pérdida de entrenamiento.

Los resultados muestran que la eliminación de una capa convolucional no mejoró significativamente el rendimiento del modelo en comparación con el modelo base. Para el *dataset* con 10 atmósferas, la pérdida en el conjunto de prueba en nanómetros es de 167, lo que representa un porcentaje de mejora del 1.02% en comparación con el error intrínseco del *dataset*. Similarmente, para los *datasets* con 5 y 2 atmósferas, los porcentajes de mejora son del 1.36% y 1.74%, respectivamente.

La eliminación de una capa convolucional parece haber reducido la capacidad del modelo para capturar características complejas, resultando en un rendimiento subóptimo. Las fluctuaciones en la pérdida de validación indican una inestabilidad en la capacidad del modelo para generalizar adecuadamente.

Dado que la eliminación de una capa convolucional no ha mostrado una mejora significativa en comparación con el modelo base y las pruebas anteriores, este cambio no será retenido en futuras pruebas. Las configuraciones de las pruebas anteriores, particularmente la cuarta prueba que aumentó el tamaño del *kernel* de la segunda capa convolucional a 4, siguen siendo las más prometedoras hasta el momento.

7.2.8. Añadido de una capa convolucional adicional

En esta octava prueba, se añadió una capa convolucional adicional al modelo original, específicamente después de las capas convolucionales existentes. La nueva capa está compuesta por 512 filtros, un *kernel_size* de 3x3, *padding* '*same'* y una función de activación ReLU. La adición de capas convolucionales adicionales puede permitir al modelo capturar características más complejas y detalladas de las imágenes, lo que podría mejorar su capacidad de predicción. Al colocar esta nueva capa al final de las capas convolucionales existentes, se busca que esta nueva capa refinada y complemente la extracción de características realizada por las capas anteriores, proporcionando una mayor profundidad y detalle a la representación de los datos antes de pasar a las capas densas.

Dataset	Error de frente de onda residual (nm)	Error de frente de onda sin corregir (nm)	Porcentaje de mejora del modelo
012_2atm	11	169	93.28%
012_5atm	47	169	72.42%
012_10atm	76	169	54.72%

Tabla 11-Resultados obtenidos tras evaluar el modelo añadiendo una capa convolucional adicional.

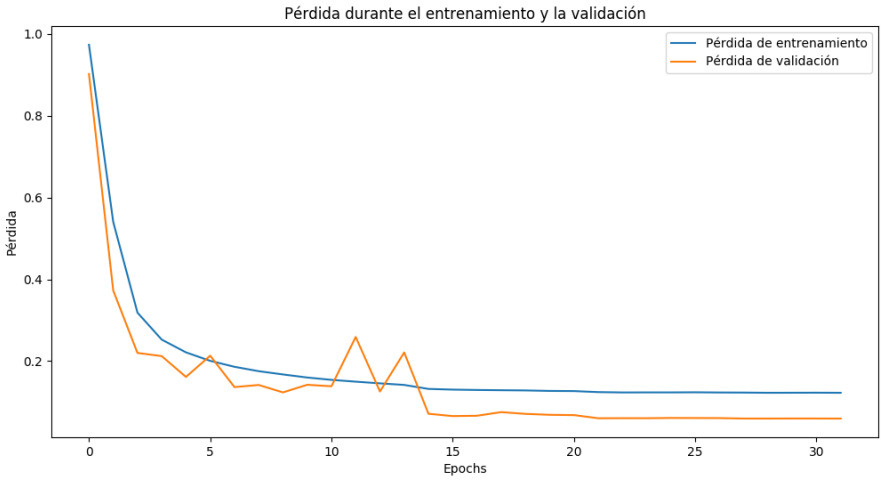

Figura 7.10- Pérdida durante el entrenamiento y la validación en la octava prueba.

La gráfica muestra una disminución rápida y constante en la pérdida tanto de entrenamiento como de validación durante las primeras 10 épocas. La pérdida de validación presenta algunas fluctuaciones menores, pero se estabiliza rápidamente y converge hacia un valor bajo similar al de la pérdida de entrenamiento.

Los resultados muestran una mejora significativa en comparación con el modelo base y otras pruebas anteriores. Para el *dataset* con 2 atmósferas, la pérdida en el conjunto de prueba en nanómetros es de 11, lo que representa un porcentaje de mejora del 93.28% en comparación con el error intrínseco del *dataset*. Similarmente, para los *datasets* con 5 y 10 atmósferas, los porcentajes de mejora son del 72.42% y 54.72%, respectivamente.

La adición de una capa convolucional adicional parece haber mejorado significativamente la capacidad del modelo para capturar características complejas, lo que se refleja en la mejora del rendimiento del modelo. La pérdida de validación muestra una

estabilización rápida, indicando que el modelo no solo está aprendiendo bien, sino que también generaliza adecuadamente a los datos de prueba.

Dado que la adición de una capa convolucional adicional ha mostrado mejoras significativas en comparación con el modelo base y todas las pruebas anteriores, se considerará retener este cambio en futuras pruebas. La estabilización rápida y la mejora en los porcentajes de mejora del modelo sugieren que esta modificación es altamente beneficiosa para el rendimiento del modelo.

7.2.9. Añadido de una segunda capa convolucional adicional

Al igual que en la prueba anterior, la capa se ha añadido en último lugar, buscando continuar con la sensacional mejora conseguida en la prueba anterior. La nueva capa está compuesta por 1024 filtros, un *kernel_size* de 3x3, *padding 'same'* y una función de activación ReLU.

Dataset	Error de frente de onda residual (nm)	Error de frente de onda sin corregir (nm)	Porcentaje de mejora del modelo
012_2atm	9,6	169	94.34%
012_5atm	43	169	74.77%
012_10atm	74	169	57.04%

Tabla 12-Resultados obtenidos tras añadir una segunda capa convolucional adicional.

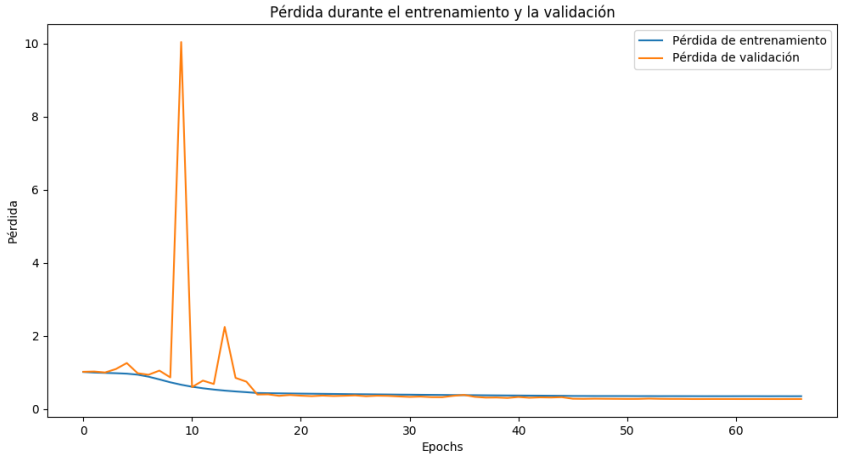

Figura 7.11- Pérdida durante el entrenamiento y la validación en la novena prueba.

La gráfica muestra una disminución rápida y constante en la pérdida tanto de entrenamiento como de validación durante las primeras 10 épocas. Sin embargo, se observan fluctuaciones significativas en la pérdida de validación, con picos notables alrededor de las épocas 10 y 20. La pérdida de validación tiende a estabilizarse después de la época 30, convergiendo hacia un valor bajo similar al de la pérdida de entrenamiento.

Los resultados muestran una mejora significativa en comparación con el modelo base y otras pruebas anteriores. Para el *dataset* con 2 atmósferas, la pérdida en el conjunto de prueba en nanómetros es de 9.6, lo que representa un porcentaje de mejora del 94.34% en comparación con el error intrínseco del *dataset*. Similarmente, para los *datasets* con 5 y 10 atmósferas, los porcentajes de mejora son del 74.77% y 57.04%, respectivamente.

La adición de dos capas convolucionales adicionales parece haber mejorado significativamente la capacidad del modelo para capturar características complejas, lo que se refleja en la mejora del rendimiento del modelo. Aunque hay fluctuaciones en la pérdida de validación, el modelo logra estabilizarse y generalizar bien a los datos de prueba.

Dado que la adición de dos capas convolucionales adicionales ha mostrado mejoras significativas en comparación con el modelo base y todas las pruebas anteriores, se considerará retener este cambio en futuras pruebas. La mejora en los porcentajes de mejora del modelo sugiere que esta modificación es altamente beneficiosa para el rendimiento del modelo.

7.2.10. Añadido de una tercera capa convolucional adicional

En esta décima prueba, se añadió una tercera capa convolucional adicional al modelo original, colocada específicamente al final de las capas convolucionales existentes, incluidas las dos anteriores añadidas. La nueva capa está compuesta por 2048 filtros, un *kernel_size* de 3x3, *padding* '*same'* y una función de activación ReLU La justificación para este cambio es que situar la nueva capa al final permite que el modelo refine aún más las características extraídas por las capas previas y así intentar mejorar la capacidad de predicción del modelo. Colocar la capa adicional al final mantiene la estructura lógica del modelo, permitiendo que cada capa sucesiva refine y construya sobre las características extraídas antes de pasar a las capas densas.

Dataset	Error de frente de onda residual (nm)	Error de frente de onda sin corregir (nm)	Porcentaje de mejora del modelo
012_2atm	19	169	88.55%
012_5atm	51	169	69.67%
012_10atm	78	169	53.52%

Tabla 13-Muestra de los resultados obtenidos tras añadir una tercera capa convolucional adicional.

Los resultados muestran una mejora significativa en comparación con el modelo base y otras pruebas anteriores. Para el *dataset* con 2 atmósferas, la pérdida en el conjunto de prueba en nanómetros es de 19, lo que representa un porcentaje de mejora del 88.55% en comparación con el error intrínseco del *dataset*. Similarmente, para los *datasets* con 5 y 10 atmósferas, los porcentajes de mejora son del 69.67% y 53.53%, respectivamente.

La adición de tres capas convolucionales adicionales parece haber mejorado significativamente la capacidad del modelo para capturar características complejas, lo que se refleja en la mejora del rendimiento del modelo. La pérdida de validación muestra una estabilización rápida, indicando que el modelo no solo está aprendiendo bien, sino que también generaliza adecuadamente a los datos de prueba.

Aunque la adición de tres capas convolucionales adicionales ha mostrado mejoras significativas en comparación con el modelo base, los resultados no superan a los obtenidos

en la prueba anterior, donde se añadieron dos capas convolucionales adicionales. Por lo tanto, este cambio no será el que figure con los mejores resultados. La configuración de la novena prueba sigue siendo la más prometedora hasta el momento.

Es posible que los resultados no sean óptimos debido a la cantidad de parámetros que se utilizan en el modelo con tres capas convolucionales adicionales. La complejidad incrementada puede haber llevado a un mayor riesgo de sobreajuste, afectando la capacidad del modelo para generalizar bien a los datos de validación.

La última capa convolucional tiene los siguientes detalles:

- Tamaño del kernel: 3x3
- Número de filtros: 2048
- Profundidad de la entrada: 1024 (esto proviene de la salida de la capa anterior)

Los parámetros para una capa convolucional se calculan como:

$$N\ de\ parametros = \begin{pmatrix} ancho\ del\ kernel\ \times\ alto\ del\ kernel \\ \times\ profundidad\ de\ entrada\ +\ 1 \end{pmatrix} \times número\ de\ filtros \quad (17)$$

Aquí, el "+1" corresponde al *bias term* (término de sesgo) para cada filtro. Aplicando estos valores:

$$N\ de\ parametros = (3 \times 3 \times 1024 + 1) \times 2048 \quad (18)$$

Por lo tanto, la última capa convolucional añade 18.876.416 parámetros, lo que refleja la complejidad y capacidad adicional introducida para mejorar el rendimiento del modelo. Y lo que haría que el modelo finalmente tuviese más de 25 millones de parámetros.

7.3. RECOPILACIÓN DE RESULTADOS

En esta sección, se presenta un resumen comparativo de los resultados obtenidos a lo largo de las diversas pruebas realizadas.

La mejor configuración de modelo se obtuvo en la novena prueba, donde se añadieron dos capas convolucionales adicionales. Esta configuración mostró la mayor mejora en el rendimiento del modelo en comparación con todas las demás pruebas.

La peor configuración de modelo se obtuvo en la séptima prueba, donde se eliminó una capa convolucional. Esta configuración mostró una mejora mínima en el rendimiento del modelo, destacándose como la menos efectiva.

		Porcentaje de mejora		
ID	**Prueba sobre el modelo**	**012_2atm**	**012_5atm**	**012_10atm**
BASE	Uso de Relu	55.79%	45.35%	34.88%
1	Uso de LeakyRelu	41.23%	32.72%	24.62%
2	Normalización "Yea-Johnson"	61.22%	48.16%	35.97%
3	Kernel primera convolucional 5x5	70.42%	55.59%	41.89%
4	Kernel segunda convolucional 5x5	81.29%	62.42%	46.14%
5	Dropout Alto y batch_size bajado (8)	56.18%	44.51%	33.83%
6	1 capa Dense Mas	76.61%	58.03%	42.93%
7	1 capa convolucional menos	1.7366%	1.3575%	1.02%
8	1 capa convolucional más (1 más)	93.28%	72.42%	54.72%
9	1 capa convolucional más (2 más)	94.02%	74.78%	57.04%
10	1 capa convolucional más (3 más)	88.55%	69.67%	53.53%

Tabla 14- Resumen de los resultados con sus respectivos porcentajes de mejora.

8. Conclusiones y trabajos futuros

A lo largo de este proyecto, he aprendido significativamente sobre la aplicación de redes neuronales convolucionales para la corrección de turbulencias atmosféricas en imágenes ópticas. El uso del sistema WOMBAT para la obtención de datos y la implementación de diversas técnicas de redes neuronales ha permitido explorar y evaluar múltiples enfoques para mejorar la calidad de las imágenes afectadas por turbulencias.

Los resultados obtenidos muestran que ciertas modificaciones en la arquitectura y configuración del modelo base pueden tener un impacto significativo en la mejora de la corrección de turbulencias. Por ejemplo, el aumento del tamaño del *kernel* en las capas convolucionales y la adición de capas convolucionales adicionales han demostrado ser particularmente efectivos. La mejor configuración encontrada en el experimento con dos capas convolucionales adicionales logró una mejora del 94.02% en condiciones de 2 capas atmosféricas, 74.78% en condiciones de 5 capas y 57.04% en condiciones de 10.

Por otro lado, algunas modificaciones, como la eliminación de una capa convolucional, resultaron en una disminución drástica del rendimiento, con mejoras marginales que no superaron el 2%. Estos hallazgos subrayan la importancia de una configuración adecuada y la complejidad inherente en el diseño de modelos de CNN para aplicaciones específicas como la corrección de turbulencias.

La importancia del trabajo radica en su contribución al campo de la óptica adaptativa y la corrección de turbulencias atmosféricas, proporcionando una base sólida para futuras investigaciones y desarrollos. El uso de CNNs ha mostrado ser una herramienta poderosa y flexible para abordar problemas complejos en el procesamiento de imágenes, y los resultados obtenidos en este proyecto confirman su potencial.

En cuanto a trabajos futuros, la investigación puede avanzar en varias direcciones a partir de los hallazgos de este proyecto. Algunas posibles líneas de investigación futura incluyen:

- **Optimización de hiperparámetros**: Continuar con la búsqueda de la mejor configuración de hiperparámetros para las CNNs, explorando técnicas más avanzadas de optimización y validación cruzada para afinar los modelos.
- **Implementación en tiempo real**: Investigar la posibilidad de implementar los modelos desarrollados en sistemas de corrección de turbulencias en

tiempo real, optimizando la eficiencia computacional y reduciendo la latencia, dado que los sistemas de óptica adaptativa han de ser rápidos para corregir inmediatamente las diferentes variaciones de la atmosfera.

- **Ampliación de conjuntos de datos**: Ampliar y diversificar los conjuntos de datos utilizados, incluyendo diferentes condiciones atmosféricas y tipos de turbulencias, para mejorar la robustez y generalización de los modelos.
- **Integración con otros sistemas de óptica adaptativa**: Explorar la integración de las CNNs con otros sistemas de óptica adaptativa para crear soluciones híbridas que combinen las ventajas de múltiples enfoques.

En resumen, el trabajo realizado ha demostrado el potencial de las CNNs para la corrección de turbulencias atmosféricas y ha abierto múltiples vías para futuras investigaciones. La continua exploración y desarrollo en este campo no solo mejorará la calidad de las observaciones ópticas, sino que también contribuirá al avance de la tecnología de procesamiento de imágenes en general.

Referencias

[1] V. Lakshminarayanan y A. Fleck, «Zernike polynomials: a guide», *J Mod Opt*, vol. 58, n.º 7, pp. 545-561, abr. 2011, doi: 10.1080/09500340.2011.554896.

[2] Y. Mejía Barbosa, M. Barbosa, y Y. El, «El frente de onda y su representación con polinomios de Zernike», 2011. [En línea]. Disponible en: https://ciencia.lasalle.edu.co/svo

[3] R. Davies y M. Kasper, «Adaptive optics for astronomy», *Annu Rev Astron Astrophys*, vol. 50, n.º Volume 50, 2012, pp. 305-351, sep. 2012, doi: 10.1146/ANNUREV-ASTRO-081811-125447/CITE/REFWORKS.

[4] R. K. Tyson y B. W. Frazier, *Principles of Adaptive Optics*. CRC Press, 2022. doi: 10.1201/9781003140191.

[5] T. G. Schneider, «Astronomical Adaptive Optics», *Dr. Dobb's Journal*, vol. 29, n.º 4, pp. 24-29, abr. 2004, doi: 10.1086/684512.

[6] R. Davies y M. Kasper, «Adaptive optics for astronomy», *Annu Rev Astron Astrophys*, vol. 50, n.º Volume 50, 2012, pp. 305-351, sep. 2012, doi: 10.1146/ANNUREV-ASTRO-081811-125447/CITE/REFWORKS.

[7] A. P. Wong *et al.*, «Predictive control for adaptive optics using neural networks», *J Astron Telesc Instrum Syst*, vol. 7, n.º 01, feb. 2021, doi: 10.1117/1.JATIS.7.1.019001.

[8] W. S. Mcculloch y W. Pitts, «A logical calculus of the ideas immanent in nervous activity», *Bulletin of mathematical biophysics*, vol. 5, 1943, Accedido: 12 de julio de 2024. [En línea]. Disponible en: https://link.springer.com/article/10.1007/BF02478259

[9] F. Rosenblatt, «The perceptron: A probabilistic model for information storage and organization in the brain», *Psychol Rev*, vol. 65, n.º 6, pp. 386-408, nov. 1958, doi: 10.1037/H0042519.

[10] A. Krizhevsky, I. Sutskever, y G. E. Hinton, «ImageNet Classification with Deep Convolutional Neural Networks», *Adv Neural Inf Process Syst*, vol. 25, 2012.

[11] Y. Lecun, Y. Bengio, y G. Hinton, «Deep learning», *Nature 2015 521:7553*, vol. 521, n.º 7553, pp. 436-444, may 2015, doi: 10.1038/nature14539.

[12] J. Osborn *et al.*, «Open-loop tomography with artificial neural networks on CANARY: on-sky results», *Monthly Notices of the Royal Astronomy Society - MNRAS*, vol. 441, pp. 2508-2514, 2014, doi: 10.1093/mnras/stu758.

[13] S. Ruder, «An overview of gradient descent optimization algorithms», 2016, Accedido: 29 de mayo de 2024. [En línea]. Disponible en: http://caffe.berkeleyvision.org/tutorial/solver.html

[14] Y. Guo *et al.*, «Adaptive optics based on machine learning: a review», *Opto-Electronic Advances*, vol. 5, n.º 7, pp. 200082-1, 2022, doi: 10.29026/OEA.2022.200082.

[15] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, y E. Muharemagic, «Deep learning applications and challenges in big data analytics», *J Big Data*, vol. 2, n.º 1, pp. 1-21, dic. 2015, doi: 10.1186/S40537-014-0007-7/METRICS.

[16] M. Gupta, K. Rajnish, y V. Bhattacharjee, «Impact of Parameter Tuning for Optimizing Deep Neural Network Models for Predicting Software Faults», *Sci Program*, vol. 2021, pp. 1-17, 2021, doi: 10.1155/2021/6662932.

[17] N. Srivastava, G. Hinton, A. Krizhevsky, y R. Salakhutdinov, «Dropout: A Simple Way to Prevent Neural Networks from Overfitting», *Journal of Machine Learning Research*, vol. 15, pp. 1929-1958, 2014.

[18] K. O'Shea y R. Nash, «An Introduction to Convolutional Neural Networks», *Int J Res Appl Sci Eng Technol*, vol. 10, n.º 12, pp. 943-947, nov. 2015, doi: 10.22214/ijraset.2022.47789.

[19] Matt. Pharr y Randima. Fernando, «GPU gems 2 : programming techniques for high-performance graphics and general-purpose computation», p. 814, 2005, Accedido: 13 de mayo de 2024. [En línea]. Disponible en: https://dl.acm.org/doi/abs/10.5555/1062395

[20] «CUDA Toolkit - Free Tools and Training | NVIDIA Developer». Accedido: 10 de julio de 2024. [En línea]. Disponible en: https://developer.nvidia.com/cuda-toolkit

[21] A. Munshi, «The OpenCL specification», *2009 IEEE Hot Chips 21 Symposium, HCS 2009*, pp. 11-314, may 2016, doi: 10.1109/HOTCHIPS.2009.7478342.

[22] P. Du, R. Weber, P. Luszczek, S. Tomov, G. Peterson, y J. Dongarra, «From CUDA to OpenCL: Towards a performance-portable solution for multi-platform GPU programming», *Parallel Comput*, vol. 38, n.º 8, pp. 391-407, ago. 2012, doi: 10.1016/J.PARCO.2011.10.002.

[23] M. Abadi y et. al, «TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems». Accedido: 2 de julio de 2024. [En línea]. Disponible en: https://dblp.uni-trier.de/rec/journals/corr/AbadiABBCCCDDDG16.html?view=bibtex

[24] D. Gyawali, A. Regmi, A. Shakya, A. Gautam, y S. Shrestha, «Comparative Analysis of Multiple Deep CNN Models for Waste Classification». Accedido: 13 de mayo de 2024. [En línea]. Disponible en: http://arxiv.org/abs/2004.02168

[25] J. P. O. Li *et al.*, «Digital technology, tele-medicine and artificial intelligence in ophthalmology: A global perspective», *Prog Retin Eye Res*, vol. 82, p. 100900, may 2021, doi: 10.1016/J.PRETEYERES.2020.100900.

[26] H. Yang, C. G. Gutierrez, N. A. Bharmal, y F. J. De Cos Juez, «Projected Pupil Plane Pattern (PPPP) with artificial neural networks», *Mon Not R Astron Soc*, vol. 487, n.º 1, pp. 1480-1487, jul. 2019, doi: 10.1093/MNRAS/STZ1362.

[27] S. H. Chan y N. Chimitt, «Computational Imaging Through Atmospheric Turbulence», *Foundations and Trends® in Computer Graphics and Vision*, vol. 15, n.º 4, pp. 253-508, oct. 2023, doi: 10.1561/0600000103.

[28] H. Yang, C. G. Gutierrez, N. A. Bharmal, y F. J. De Cos Juez, «Projected Pupil Plane Pattern (PPPP) with artificial neural networks», *Mon Not R Astron Soc*, vol. 487, n.º 1, pp. 1480-1487, jul. 2019, doi: 10.1093/MNRAS/STZ1362.

[29] M. Riani, A. C. Atkinson, y A. Corbellini, «Automatic robust Box–Cox and extended Yeo–Johnson transformations in regression», *Stat Methods Appt*, vol. 32, n.º 1, pp. 75-102, mar. 2023, doi: 10.1007/S10260-022-00640-7/FIGURES/10.