Universidad Autónoma de Querétaro
Facultad de Ingeniería

Reconstrucción de modos espaciales de luz a partir de redes neuronales profundas

Tesis

Que como parte de los requisitos para obtener el Grado de
Licenciado en Ingeniería Física

Presenta:
Bruno Iván Salgado Molina

Dirigido por:
Dr. Jorge Luis Domínguez Juárez
Co-Director:
Dr. Mario Alan Quiroz Juárez

Querétaro, Qro. a abril de 2024

Universidad Autónoma de Querétaro
Facultad de Ingeniería
Licenciatura en Ingeniería Física

Reconstrucción de modos espaciales de luz a partir de redes neuronales profundas

Tesis

Que como parte de los requisitos para obtener el Grado de

Licenciado en Ingeniería Física

Presenta:

Bruno Iván Salgado Molina

Dirigido por:

Dr. Jorge Luis Domínguez Juárez

Co-Director:

Dr. Mario Alan Quiroz Juárez

SINODALES

Dr. Jorge Luis Domínguez Juárez Presidente	_____ Firma
Dr. Mario Alan Quiroz Juárez Secretario	_____ Firma
Dra. María Lucero Gómez Herrera Vocal	_____ Firma
Dr. Marco Antonio Aceves Fernández Suplente	_____ Firma

Campus Aeropuerto, Querétaro, Qro.

Abril de 2024

México

Resumen

Los haces de luz estructurada que llevan momento angular orbital (OAM) han sido utilizados para codificar información en el área de la comunicación óptica usando protocolos que destacan por su robustez. Sobresalen los haces de Laguerre-Gauss por dichas propiedades de OAM que les otorga alta dimensionalidad y que le permite la implementación de protocolos que transmiten grandes cantidades de información. Recientemente, la pérdida de su perfil espacial al propagarse en medios dispersivos ha sido aprovechada como mecanismo de encriptación natural, y se han utilizado redes neuronales multicapa como mecanismo de desencriptación por su capacidad para reconocer y clasificar patrones. En esta tesis se busca optimizar el proceso de desencriptación de los modos espaciales de luz al eliminar el pre-procesamiento de las distribuciones de intensidad, implementando redes neuronales convolucionales (CNN) por su capacidad para extraer características de los patrones de luz de forma automática, siendo invariantes a traslaciones, escalamientos, rotaciones y pequeñas distorsiones. Este es un paso importante para el desarrollo de las tecnologías de comunicación cuántica ya que la aplicabilidad de los haces de luz estructurada con OAM se puede extender a medios dispersivos como las fibras ópticas multimodo comerciales o comunicaciones en espacio libre que, aunque no conservan estos perfiles espaciales, son de las más usadas en la industria de la comunicación.

(**Palabras clave:** Redes neuronales convolucionales, haces Laguerre-Gauss, procesamiento de imágenes)

ABSTRACT

Structured light beams that carry orbital angular momentum (OAM) have been used to develop robust protocols to encrypt information in the optical communication area. Among them, Laguerre-Gaussian beams stand out because of their OAM properties which confer their high dimensionality, and permit the implementation of protocols that transmit large amounts of information. Recently, the loss of their spatial profile during its propagation through dispersive media has been used as a mechanism for natural encryption, and multilayered neural networks have been utilized for pattern recognition and classification. This project seeks to optimize the process for de-encryption of the spatial modes of light through the elimination of pre-processing of the intensity distributions by implementing convolutional neural networks (CNN) because of their capacity to automatically extract characteristics from the light patterns, being less sensible to translations, zooming, rotations and minor distortions. This is an essential step for the development of quantum communication technologies since the applicability of structured light beams with OAM can be extended to dispersive media such as commercial multimode optic fibers and free-space communications that, even though they do not conserve these spatial profiles, are most used in the communication industry.

(**Key words:** Convolutional neural networks, Laguerre-Gauss beams, image processing)

Con profundo agradecimiento y amor, dedico esta tesis

A mi mamá y a mi papá,

A Frida, mi novia

Y a quienes creyeron en mí

AGRADECIMIENTOS

En la realización de la presente tesis se vieron involucradas en mayor o menor medida varias personas e instituciones a las cuales les expreso mi más sincero agradecimiento:

Al Dr. Mario Alan Quiroz Juárez, por permitirme trabajar bajo su tutela y compartirme sus conocimientos en redes neuronales, por tomarse el tiempo y tener paciencia al aclarar todas mis dudas y al corregirme cuando fue necesario, y por todo su apoyo y guía en la etapa final de mi carrera.

Al Dr. Jorge Luis Domínguez Juárez, por guiarme en la comprensión del comportamiento de los láseres y las fibras ópticas, por mostrarme lo interesante que es la fotónica y la ciencia experimental, por inspirarme con el entusiasmo con el que siempre habla de sus investigaciones e ideas.

A la Dra. María Lucero Gómez Herrera, por su continuo apoyo como sinodal y coordinadora, por sus comentarios para mejorar el contenido del documento y por siempre decirnos que no nos rindamos con hacer la tesis.

Al Dr. Marco Antonio Aceves Fernández, por tomarse el tiempo de revisar los detalles del proyecto y compartir sus observaciones y comentarios que, sin duda, enriquecieron esta tesis.

A la Facultad de Ingeniería de la Universidad Autónoma de Querétaro (UAQ) y a la propia UAQ, por brindarme los medios para estudiar una ciencia tan maravillosa como es la física. A cada docente de la carrera de Ingeniería Física que me compartió sus conocimientos.

A la Dirección General de Asuntos del Personal Académico (DGAPA) UNAM, por el apoyo otorgado a través del Programa de Apoyo a Proyectos de Investigación e Innovación Tecnológica bajo el proyecto TA101023.

A mis padres, Gabriela y Juan Manuel, quienes me han guiado y apoyado incondicionalmente en cada momento, he llegado hasta aquí gracias a ellos y no hay palabras suficientes para agradecer lo que han hecho por mí. A mis hermanas por todo su apoyo y su cariño.

A mi novia, Frida Abascal, por el gran trabajo al hacer los fondos de la GUI y el diseño del poster de esta tesis, por darme ánimos cuando no creía en mí mismo, por inspirarme a hacer cosas nuevas y a no tenerle miedo a la vida, pero sobre todo por ser mi mejor amiga y mi más grande apoyo todos estos años. Te amo.

Finalmente, a mis amigos, por su apoyo en las buenas y en las malas, por alentarme a superarme y por aconsejarme cuando lo necesité.

INDICE

INDICE DE TABLAS

INDICE DE FIGURAS

ABREVIATURAS Y SIGLAS

CNN: Siglas en inglés para Red Neuronal Convolucional (Convolutional Neural Network).

D^2NNs: Siglas en inglés para Redes Neuronales Profundas Difractivas (Deep Diffractive Neural Nets).

ECM: Siglas para Error Cuadrático Medio

GUI: Siglas en inglés para Interfaz Gráfica de Usuario (Graphical User Interface).

ICN-UNAM: Siglas para Instituto de Ciencias Nucleares de la Universidad Nacional Autónoma de México.

MLP: Siglas en inglés para Perceptrón Multicapa (Multi-Layered Perceptron).

Modo HG: Modo de Hermite-Gauss.

Modo LG: Modo de Laguerre-Gauss

OAM: Siglas en inglés para Momento Orbital Angular (Orbital Angular Momentum).

ReLU: Abreviación para Rectificador Lineal Unitario

RGB: Siglas en inglés para el modelo de color Rojo-Verde-Azul (Red-Green-Blue)

RNA: Siglas de Red Neuronal Artificial.

SAM: Siglas en inglés para Momento Angular de Espín (Spin Angular Momentum).

Soluciones LP: Soluciones Linealmente Polarizadas.

SVM: Siglas en inglés para Máquina de Soporte Vectorial (Support Vector Machine).

TLU: Siglas en inglés para Unidad Lógica de Umbral (Threshold Logic Unit).

I. INTRODUCCIÓN

En las últimas décadas los haces de luz espacialmente estructurados han ganado interés por sus potenciales aplicaciones en diversas áreas como la microscopía e imagenología o el confinamiento óptico para la manipulación de partículas microscópicas (Páez López, 2017). Sin embargo, una de las aplicaciones más prometedoras es el uso de haces de luz que llevan momento angular orbital, cambiando su distribución espacial a un patrón de hélice, en el campo de las comunicaciones ópticas (Willner et al., 2015).

Particularmente, resultan interesantes los haces de Laguerre-Gauss por sus propiedades de momento angular orbital. En este tipo de haces, la forma helicoidal se induce por una dependencia de fase del tipo $e^{il\phi}$, donde l es el número cuántico de momento angular orbital, y ϕ es el ángulo azimutal. Esto les permite almacenar una gran cantidad de información en un solo fotón, abriendo la posibilidad de desarrollar protocolos de encriptación de comunicación más robustos que los convencionales (Allen, Padgett y Babiker,1999; Bhusal et al., 2021). Aunado a esto, se ha demostrado que, en general, los haces de luz estructurada permiten obtener mayores niveles de seguridad contra espías, característica fundamental para la comunicación segura (Mirhosseini et al., 2015; Malik et al., 2012; Gröblacher et al., 2006; Bourennane et al., 2002).

Bajo condiciones controladas, dichos haces pueden transmitirse de un punto a otro sin perder información, pero en situaciones reales, como el espacio libre con turbulencias (Jurado-Navas et al., 2015) o en las fibras ópticas multimodo comerciales (Lollie et al., 2022; Cozzolino et al, 2019) los haces de Laguerre-Gauss no corresponden a modos de propagación. La distribución espacial en forma helicoidal de los haces se pierde parcialmente por fenómenos como la dispersión, por lo cual su estudio se limitaba a la propagación en espacio vacío o en fibras más específicas como las vórtex, fibras con núcleo de aire (Cozzolino et al., 2019) o fibras de múltiples núcleos (Ding et al., 2017; Cañas et al., 2017).

En este sentido, investigaciones recientes han explorado alternativas para poder mejorar la aplicabilidad de los haces estructurados a través del uso de elementos

ópticos adaptativos (Jurado-Navas et al., 2015) y mecanismos de Machine Learning como: (i) Máquinas de Soporte Vectorial (SVM por sus siglas en inglés) para decodificar información proveniente de haces de vórtice vectorial distorsionados por caminatas cuánticas (Giordani et al., 2020) y medios turbulentos (Sun et al., 2019), (ii) redes multicapa con preprocesamiento para decodificar haces LG distorsionados por fibra óptica (Lollie et al., 2022), (iii) redes convolucionales para detección de modos de Hermite-Gauss (Hofer, Jones, Goedert y Dragone, 2019), reconstrucción de modos LG superpuestos con fluctuaciones por turbulencia (Bushal et al., 2022) y reconocimiento de cargas topológicas de modos OAM (Liu, Yan, Liu, y Chen, 2019), e inclusive métodos compuestos como (iv) redes neuronales profundas difractivas (D^2NNs) en el procesamiento de haces de vórtice para identificación, generación y conversión de modos híbridos.

Es de especial importancia notar que en el trabajo de Lollie et al. (2022) se mostró que es posible usar el fenómeno de dispersión como método de encriptación natural y recuperar la información espacial perdida en el trayecto empleando redes neuronales, las cuales reconocen las distribuciones de intensidad de los haces distorsionados y los reconstruye satisfactoriamente.

Sin embargo, la arquitectura de la red utilizada en ese trabajo consistía en una red multicapa, la cual no resulta muy eficiente para la tarea de clasificación de imágenes (Haykin, 1999). Además, las imágenes recopiladas requieren un procesamiento previo detallado antes de poder ser introducidas a la red neuronal para su clasificación.

Partiendo de las bases planteadas anteriormente, la presente tesis tiene por objetivo implementar el uso de redes convolucionales (CNN) para el reconocimiento de patrones de intensidad provenientes de haces de Laguerre-Gauss que viajan a través de un medio dispersivo. La elección del modelo de clasificación está motivada por los siguientes argumentos:

I. Las redes neuronales convolucionales presentan altos rendimientos en problemas de clasificación de imágenes, como el que se abordará en este trabajo.
II. Debido a la arquitectura de la red neuronal convolucional, la extracción de características de los patrones de intensidad es automática. Esto conduce a la invarianza ante traslaciones, escalamientos, rotaciones y pequeñas distorsiones (Haykin, 1999; LeCun, Bengio y Hinton, 2015).
III. Las propiedades de las capas de convolución permiten la anulación de los pasos de procesamiento previo de las imágenes.

Esta tarea es importante ya que, al eliminar el requerimiento de procesamiento previo de las imágenes, es posible incluir la red neuronal entrenada en algún dispositivo que reciba las imágenes recién capturadas y, sin intervención humana para procesarlas, habilite la clasificación de los modos espaciales con momento angular orbital.

II. FUNDAMENTACIÓN TEÓRICA

1. Redes neuronales

1.1. Concepto

Existen diversas formas de definir lo que es una red neuronal artificial (RNA) puesto que es un concepto que ha ido evolucionando desde la concepción del primer modelo de neurona artificial concebido por McCulloch y Pitts (1943), en donde se introdujo el concepto de las redes neuronales como máquinas que computan operaciones (Haykin, 1999; Ponce Cruz, 2010; Gurney, 2004; Yáñez Márquez, López Leyva & Aldape Pérez, 2007). Sin embargo, muchas de estas definiciones se pueden dividir por su enfoque en la similitud que tienen las RNA con las redes neuronales biológicas o por su enfoque más matemático sobre la forma de procesar la información que reciben.

Muchos autores como Rojas (1996), Navarro Pastor (1998), Haykin (1999) y Ponce Cruz (2010) concuerdan en la idea de que las redes neuronales artificiales son sistemas de procesamiento cuyas unidades, distribuidas masivamente y conectadas entre sí de forma paralela, se basan en las neuronas animales o humanas y tienen el propósito de reproducir la forma de procesar información llevada a cabo por los sistemas nerviosos tanto humanos como animales. En específico Gurney (2004) nos dice que "desde un punto de vista biológico el requisito esencial para una red neuronal es que debería intentar capturar lo que creemos que son las características esenciales de procesamiento de información de la red 'real' correspondiente" (p. 19).

Siguiendo esta línea de pensamiento Navarro Pastor (1998), citando los trabajos de Crick y Asanuma (1986), Rumelhart y McClelland (1986), Sejnowski y Rosenberg (1986), Mira (1993) y Nelson e Illingworth (1991), entre otros, destaca que para plantear los modelos conexionistas que describen a las redes neuronales se tomaron en cuenta algunas de las siguientes características del cerebro:

- Las neuronas son cerca de 6 ordenes de magnitud más lentas que los componentes computacionales convencionales. Ya que el ser humano

puede llevar a cabo tareas altamente complejas en cientos de milisegundos sería muy difícil que las neuronas trabajen por conexión en serie, en su lugar alcanzamos el éxito con un procesamiento cerebral paralelo masivo.

- El conocimiento a largo plazo se almacena en las conexiones entre neuronas, mientras que en las propias neuronas se almacena solo información a corto plazo. Lo cual implica que el conocimiento se almacena de forma distribuida pues un proceso cognitivo no se representa por una única neurona sino por un patrón de actividad que se distribuye por un gran número de neuronas simultáneamente, permitiendo que cada una de las neuronas forme una parte en la representación de un gran número de procesos cognitivos.
- La información transmitida entre neuronas es de tipo numérico pues se comunican a través de señales de excitación e inhibición.
- Luego de estudiar lesiones cerebrales se ha observado que no existe una neurona que sea esencial para la realización de algún proceso cognitivo, más bien la capacidad de procesamiento del cerebro se degrada poco a poco conforme más neuronas se destruyen sin que exista un punto crítico exacto en el cual el funcionamiento se interrumpa.

De forma similar al proceso de aprendizaje basado en la experiencia usado por los humanos, las RNA utilizan problemas resueltos para construir un sistema que pueda afrontar situaciones nuevas (Ponce Cruz, 2010) siguiendo algún mecanismo para aprender (Goodfellow, Bengio & Courville, 2016). Para las RNA, dicho mecanismo de aprendizaje implica una modulación de los pesos sinápticos (i.e. conexión entre neuronas), lo cual es una característica fundamental que dotan a las RNA de un carácter conexionista pues es en las conexiones donde se almacena el conocimiento y no en las propias neuronas (Navarro Pastor, 1998; Haykin, 1999, Gurney, 2004).

Por otro lado, una forma matemática de pensar en las neuronas es como un conjunto de funciones primitivas que entregan determinadas salidas según los

estímulos que reciban como datos de entrada (Rojas, 1996). Con ello, las RNA pueden ser concebidas como una red dinámica o grafo dirigido que consiste en nodos que representan funciones primitivas cuyas salidas dependen de la información local disponible almacenada internamente u obtenida por las conexiones a otros nodos, moduladas por los pesos sinápticos (Haykin, 1999; Ponce Cruz, 2010). En este sentido, las RNA no buscan modelar el funcionamiento del cerebro, más bien son máquinas diseñadas para alcanzar generalizaciones estadísticas de datos (Goodfellow, Bengio y Courville, 2016).

Se puede considerar que existen siete elementos que caracterizan a una RNA (Rumelhart, Hinton y McClelland, 1987):

1. Unidades de procesamiento (neuronas). Se encargan de tomar la suma ponderada de las señales de entrada usando los pesos sinápticos y un sesgo (campo local inducido de la neurona), y transformarla mediante el enlace de activación que genera la salida (Haykin, 1999). Se pueden clasificar en neuronas de entrada, salida u ocultas.
2. La función de salida de una neurona. Esta dicta la magnitud de sus salidas basándose en su activación. En muchos casos, la salida es igual a la activación ($f(x) = x$), aunque en otros se aplica una función umbral para que una neurona no afecte a sus vecinos a menos que su activación supere un cierto valor crítico. En algunos otros casos, se utilizan funciones estocásticas que hacen que la salida dependa de las activaciones de forma probabilística.
3. El patrón de conectividad. Es la forma en la que cada neurona se vincula con el resto de la red y que determina en gran medida como esta responde a los estímulos de entrada y el tipo de procesos que puede llegar a representar la red en conjunto. Este patrón se representa por una matriz de pesos sinápticos, los cuales especifican la fuerza de la conexión de cada par de nodos de la red. Pensando en la similitud con la versión biológica, las conexiones excitatorias son de signo positivo y las inhibitorias son de signo negativo.

Una parte importante del patrón de conectividad es la arquitectura, la cual determina la forma en la que la información fluye a través de la red. Existen diversas topologías que pueden tener las RNA, pero dos de las clasificaciones más utilizadas son (Ponce Cruz, 2010):

- Redes de propagación hacia delante (feed-forward): La transmisión de la información desde las entradas hasta las salidas sigue una única dirección, pasando a través de varias capas de unidades sin ninguna conexión de retroalimentación.
- Redes recurrentes: Incluyen conexiones de retroalimentación, lo que puede conducir a un proceso evolutivo hacia un estado estable en el que las neuronas no experimenten cambios en su estado de activación.

4. Regla de propagación. Es una función que determina la entrada efectiva o campo local inducido de la neurona a partir de las entradas externas y los pesos sinápticos del patrón de conectividad. Usualmente esta es solo la suma ponderada de las entradas, pero para representar interacciones más complejas puede ser diferente.
5. Función de activación. Nos dice como el campo local inducido u nos devuelve un cierto estado de activación de la neurona (que muchas veces será la salida de la neurona). Las funciones más comunes son (Cheng y Titterington, 1994; Haykin, 1999; Dinamarca, 2018):

    - Función signo: Es una de las diversas funciones que proporciona salidas binarias, en este caso siendo ± 1.

    $$f(u) = sgn(u) = \begin{cases} -1 si u < 0 \\ 1 si u \geq 0 \end{cases} \quad (1)$$

    - Función umbral: Una de las funciones binarias más usadas, también se le suele llamar función de Heaviside, proporciona la salida binaria típica 1/0.

    $$f(u) = \frac{sgn(u-\theta)+1}{2} = H_\theta(u) = \begin{cases} 0 si u < \theta \\ 1 si u \geq \theta \end{cases} \quad (2)$$

- Unidad Lineal Rectificada (ReLU): Es una función que amortigua o anula todas las entradas menores a un cierto umbral (usualmente el cero). También se conoce como función rampa.

$$f(u) = \begin{cases} usiu < a \\ \beta usiu \geq a \end{cases} \quad (3)$$

donde $\beta \leq 1$, y $a \in R$ es el valor umbral para el cual empieza a haber una correspondencia lineal.

- Función lineal mixta: Una función de salida continua ya sea en el rango $(-1,1)$ o $(0,1)$ según se defina.

$$f(u) = \begin{cases} 0o - 1siu \leftarrow \theta \\ axsi - \theta < u < \theta \\ 1siu \geq \theta \end{cases} \quad (4)$$

- Función sigmoidal: Una función de salida continua en el rango $(0,1)$ cuya gráfica tiene forma de "S". Es una de las más utilizadas en las RNA porque su derivada es continua y está definida para todo u, además de que es una función siempre creciente que balancea el comportamiento lineal y no lineal. Un ejemplo sería la función logística:

$$f(u) = \frac{1}{1 + e^{-au}} \quad (5)$$

donde el parámetro a determina lo pronunciada que será la función.

- Función tangente hiperbólica: Similar a la logística, pero con salidas que se encuentra en el rango $(-1,1)$:

$$f(u) = tanh\,(u) = \frac{e^{u} - e^{-u}}{e^{u} + e^{-u}} \quad (6)$$

- Función Gaussiana: Función continua que se usa en ocasiones que requieren dar mayor importancia a campos locales inducidos que se encuentren cerca de un valor determinado.

$$f(u) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{u^2}{2\sigma^2}} \tag{7}$$

donde σ es un parámetro que nos permite controlar el ancho de la gaussiana (Ponce Cruz, 2010).

- Función binaria probabilística (Little, 1974): Usada para modelos estocásticos de neuronas, tomará valores excitativos (1) o inhibitorios (-1) según una determinada probabilidad $P(u)$.

$$f(u) = \begin{cases} 1 \ \ con \ \ probabilidad \ \ P(u) \\ -1 \ \ con \ \ probabilidad \ \ 1 - P(u) \end{cases} \tag{8}$$

donde una forma habitual de escoger $P(u)$ está dada por la función sigmoide

$$P(u) = \frac{1}{1 + e^{-u/T}} \tag{9}$$

donde T se conoce como *pseudotemperatura* y se utiliza para controlar la incertidumbre de la activación.

6. Regla de aprendizaje. Es el método utilizado por la red para poder alterar el patrón de conectividad según se gana experiencia, almacenando el conocimiento que se adquiere. Para ello, se requiere el uso de métricas de desempeño, como la exactitud o la tasa de error, que nos muestren el desempeño de la red en conjuntos de datos de evaluación independientes a los que se usen para el propio entrenamiento (Dinamarca, 2018). Esto puede implicar dos modificaciones:
    - La alteración de la arquitectura al desarrollar o destruir conexiones entre neuronas.
    - La modulación de los pesos sinápticos.

De ellas el segundo tipo es el más desarrollado, siendo la retropropagación del error una de las reglas de aprendizaje más conocidas.

7. Ambiente. Es el entorno en el que opera y se desarrolla la red neuronal. Se caracteriza por un conjunto de posibles patrones de entrada que alimentan a la red y en algunas ocasiones incluye señales de error externas a la RNA.

De entre todas estas características, Rojas (1996) y Ponce Cruz (2010) destacan que los elementos más importantes a considerar para las RNA son los modelos de las neuronas (influyen los puntos 1, 2, 4 y 5 de la lista), la topología de la red (punto 3 de la lista) y el algoritmo de aprendizaje (punto 6 de la lista).

Como es señalado por Rumelhart, Hinton y McClelland (1987), Haykin (1999) y Ponce Cruz (2010), algunas de las ventajas que tienen las RNA respecto a la programación tradicional son que:

- Sintetizan algoritmos en el proceso de aprendizaje que de otro modo podrían ser muy extensos.
- No requieren que el usuario tenga conocimiento de los detalles matemáticos de cada problema.
- Salvo por las RNA más simples, pueden solucionar tanto problemas lineales como no lineales.
- Tienen tolerancia a la falla de alguno de sus componentes (i.e. son robustos).
- Se pueden generalizar a nuevas entradas y dar resultados razonables.
- Son adaptativas, ya que pueden alterar la fuerza de las conexiones neuronales para adaptarse a los posibles cambios del ecosistema en el que se usen.

Mientras que algunas de sus desventajas son (Ponce Cruz, 2010):

- Para cada tarea específica hay que realizar el entrenamiento, proceso que requiere muchos datos y puede necesitar un tiempo considerable para alcanzar la arquitectura y pesos adecuados.
- Para redes de tamaño considerable resulta complicado realizar cambios específicos pues, una vez entrenadas, las RNA tienen cantidad grande de unidades y conexiones bastante intrincadas (Cheng y Titterington, 1994).

Las RNA tienen aplicaciones muy variadas que incluyen la industria, los automóviles, las finanzas y la banca, la ingeniería (electrónica, telecomunicaciones, robótica), la medicina, la investigación, entre otras (Gurney, 2004; Ponce Cruz, 2010).

1.2. Redes multicapa

La clasificación de las neuronas mencionada en el primer punto de la lista de siete elementos que caracterizan a una RNA (Rumelhart, Hinton y McClelland, 1987) nos permite dividirlas de manera general en tres tipos de capas, como se muestra en la Figura 1, según su relación con el ambiente (Nielsen, 2019):

- Capa de entrada: Está formada por todas las neuronas que reciben los datos del ambiente.
- Capa de salida: Se conforman por las neuronas que entregan el resultado final obtenido por el procesamiento de los datos de entrada al pasar por la red. El número de neuronas de esta capa corresponde con las categorías en las que podría caer una observación realizada por la RNA.

- Capas ocultas: Son aquellas que toman el papel de intermediarias entre las neuronas de la capa de entrada y las neuronas de la capa de salida, haciendo más robusto el procesamiento de la información y permitiendo la realización de tareas más complejas.

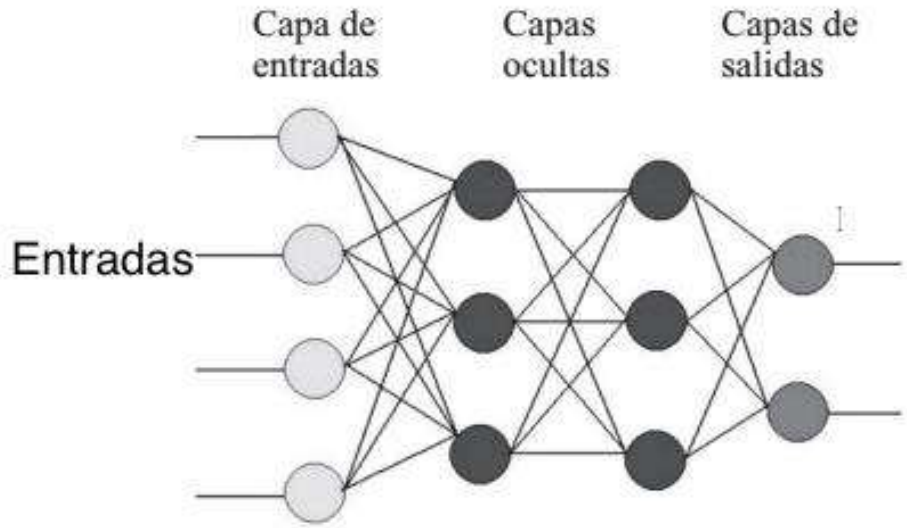

Figura 1. Esquema básico de una RNA. (Ponce Cruz, 2010).

Como ya fue mencionado con anterioridad, la alta capacidad de procesamiento que pueden alcanzar las RNA se basa en el computo paralelo masivo, aplicando el conocimiento que se obtiene de la experiencia almacenado en las conexiones entre los componentes de la red (neuronas) a través de los pesos sinápticos (Navarro Pastor, 1998). En ese sentido, es posible concebir un sin fin de RNA diferentes que varíen en la estructura de sus componentes, sus arquitecturas y sus métodos de aprendizaje. Sin embargo, es de especial interés para la investigación revisar los siguientes modelos de RNA.

1.2.1. Perceptrón simple

Basándose en el modelo de las neuronas y sistemas nerviosos como máquinas computacionales planteado por McCulloch y Pitts (1943), Rosenblatt (1957) concibió el primer modelo de aprendizaje supervisado a través del perceptrón. A grandes rasgos, Rosenblatt lo definió como un sistema electrónico que aprende a reconocer similitudes entre patrones de información de forma similar al proceso de percepción de un cerebro real. Destaca que se fundamenta en principios

probabilísticos para operar y obtiene su éxito de las propiedades de mediciones estadísticas obtenidas de un gran número de elementos de prueba. Aunque en su documento original planteó al perceptrón como un conjunto de redes más complejas a lo que se suele llamar perceptrón hoy en día (Ponce Cruz, 2010).

Lo que actualmente se conoce como perceptrón es un dispositivo que permite clasificar un conjunto de N estímulos de entrada en dos clases C_1 o C_2 (Haykin, 1999; Ponce Cruz, 2010). Dicho de otro modo, un perceptrón es un dispositivo que toma decisiones con evidencia ponderada (Nielsen, 2019).

De forma general, el perceptrón toma un conjunto de *N+1* entradas (binarias o no binarias) $x_0, x_1, x_2, \ldots, x_N$ y sus correspondientes pesos sinápticos $w_0, w_1, w_2, \ldots, w_N$ (representados por los vectores columna *X* y *W*, respectivamente), considerando que w_0 es el sesgo y $x_0 = 1$, para formar una suma ponderada como campo local inducido (Ponce Cruz, 2010; Nielsen, 2019):

$$u = \sum_{j=0}^{N} w_j \, x_j = W^T X \tag{10}$$

El cual se aplica en una función de activación binaria no lineal discreta de tipo heaviside o signo, razón por la cual algunos autores como Gurney (2004) la conocen como Unidad Lógica de Umbral (TLU por sus siglas en inglés). De este modo se puede pensar en que, el perceptrón separa el hiperespacio de atributos N-dimensional creado por cada una de las N variables de entrada, X, en dos regiones por un hiperplano definido por (Rojas, 2996; Haykin, 1999; Gurney 2004):

$$\sum_{j=0}^{N} w_j \, x_j = W^T X = 0 \tag{11}$$

En un lado del hiperplano se encuentran todos los patrones para los que el perceptrón regresa una salida de 1, mientras que aquellos clasificados como 0 se encuentran del otro lado, motivo por el cual se le conoce como un clasificador lineal (Gurney, 2004).

La condición para la correcta clasificación de los datos por un perceptrón simple es que las dos clases C_1 y C_2 deben ser linealmente separables. Esto nos dice

que los patrones a clasificar deben estar lo suficientemente separados entre sí como para asegurar que la superficie de decisión consista en un hiperplano (Haykin, 1999; Ponce Cruz, 2010).

Tomando esto en cuenta, Rosenblatt (1962), citado por Cheng y Titterington (1994, p. 15) y Haykin (1999, p. 53), demostró el teorema de convergencia de incremento fijo para el perceptrón:

> Sean los subconjuntos de vectores de entrenamiento H_1 y H_2 linealmente separables, pertenecientes a las clases C_1 y C_2, correspondientemente. Sean las entradas presentadas al perceptrón originarias de estos dos subconjuntos. Si existe un vector W_0 de pesos que resuelven el problema, el perceptrón converge luego de alguna iteración n_0, en el sentido de que
>
> $$W(n_o) = W(n_o + 1) = W(n_o + 2) = \ldots \quad (12)$$
>
> es un vector solución para $n_0 \leq n_{max}$.

Es decir, siempre que las clases sean linealmente separables, el algoritmo de entrenamiento del perceptrón convergerá a la configuración del hiperplano en un número finito de pasos.

Desde su concepción en 1957 por Ronsenblatt, sus propiedades se han analizado y autores como Gurney (2004), Ponce Cruz (2010) y Nielsen (2019) han señalado que el perceptrón simple tiene las ventajas de:

- Poder reproducir el funcionamiento de las funciones u operadores lógicos como AND, OR, NOR y NAND. Más aún, el hecho de que se puede calcular la operación NAND con un perceptrón, quiere decir que puede calcular cualquier función lógica (e.g. la suma de dos números binarios).
- Es resistente al ruido, ya que no afectan las variaciones en los valores de entrada si estas no atraviesen el valor crítico de la función umbral. Esto implica que el perceptrón simple es robusto ante la presencia de señales de entrada ruidosas o corruptas.

Por otro lado, se hallaron problemas debido a la condición de linealidad, el campo receptivo y la forma de la función de activación binaria, de los cuales se presentaron los siguientes defectos en los perceptrones simples:

- Minsky y Papert (1967), citados por Cheng y Titterington (1994), Rojas (1996) y Ponce Cruz (2010), demostraron que las funciones Booleanas de dos variables, XOR y XNOR, no pueden ser replicadas por los perceptrones simples, ya que sus salidas no son linealmente separables.
- No permite la introducción de métodos de entrenamiento que se basen en que pequeños cambios en los pesos causen correspondientemente pequeños cambios en las salidas, ya que pequeños cambios en la red producen cambio nulos o abruptos (de 0 a 1 o viceversa). Esto complica la modificación gradual de los pesos (Dinamarca, 2018).
- Minsky y Papert (1967), citados por Rojas (1996), dicen que, debido al procesamiento en paralelo, ningún perceptrón de campo receptivo limitado puede decidir si una figura geométrica está conectada o no, o si un conjunto de puntos es par o impar. Esto se debe a que las propiedades de conectividad de una figura geométrica y la paridad son globales, no pueden decidirse localmente.

Dichos problemas se arreglan en gran medida al recurrir al perceptrón de dos capas o al cambiar la función de activación de la neurona por una continua (Ponce Cruz, 2010; Goodfellow, Bengio y Courville, 2016).

1.2.2. Perceptrón multicapa

Para enfrentar las limitaciones del perceptrón simple señaladas por Minsky y Papert (1967), el siguiente paso fue introducir la estructura de RNA conocida como el perceptrón multicapa (MLP por sus siglas en inglés), pues este es un mecanismo de predicción más robusto y flexible (Cheng y Titterington, 1994).

Lo que caracteriza al perceptrón multicapa es (Cheng y Titterington, 1994; Haykin, 1999):

- La red contiene cuando menos una capa oculta de nodos (neuronas) que no son ni de entrada ni de salida. Dichos nodos solo tienen

permitido enlaces con nodos de capas consecutivas y dichos enlaces se direccionan hacia la capa de salida. En consecuencia, se dice que el MLP es una red neuronal feed-forward.

Además, son las neuronas pertenecientes a estas capas aquellas que funcionan como detectores de características de los datos de entrada, ya que estas realizan una transformación no lineal sobre el campo local inducido u a través de la función de activación $f(u)$ en un nuevo espacio conocido como el espacio de características, el cual suele simplificar la tarea de clasificación de datos.

- Cada neurona en la red cuenta con una función de activación no lineal diferenciable, ya que de otro modo el aprendizaje de la red estaría severamente limitado. Esto nos dice que el funcionamiento ya no se basa en funciones de activación binarias de tipo umbral.
- La red cuenta con un alto grado de conectividad, cuya extensión se determina por el patrón de conectividad (Rumelhart, Hinton y McClelland, 1987). Se dice que una red está completamente conectada si una neurona de cualquier capa de la red está conectada con todas las neuronas de la capa anterior.

Para poder estudiar uno de los usos más interesantes de los MLP diversos autores como Mhaskar (1993), Navarro Pastor (1998), Liu, Yang y Cai (2019), entre otros, recurren a lo que estipula el teorema de Kolgomorov:

> (...) demuestra que una función continua de varias variables puede ser representada por la superposición de funciones continuas unidimensionales de las variables de entrada originales. Cualquier función continua mapeada en una entrada de N dimensiones, $n \geq 2$, a una salida de M dimensiones puede ser implementada por una red con una capa oculta (...) (Ponce Cruz, 2010, p. 224)

En ese sentido, Navarro Pastor (1998) señala que el MLP suele incluir el uso de funciones de activación de tipo sigmoide como la logística y la tangente

hiperbólica en lugar de las funciones de tipo umbral como la función de Heaviside por tres motivos:

- Son funciones acotadas entre dos valores, uno superior y otro inferior. Esto quiere decir que los valores que toma *f(u)* nunca salen de un intervalo finito definido por la función.
- Son funciones cuyos valores *f(u)* aumentan conforme aumenta el valor del campo local inducido u. Esto quiere decir que son monótonas crecientes.
- Son funciones continuas y suaves. Es decir, no presentan discontinuidades en ellas ni sus derivadas, lo cual es fundamental para la implementación de algoritmos que se basan en derivadas para optimizar el cambio de pesos sinápticos, como los basados en el gradiente.

Lo cual nos asegura la existencia de una configuración de la RNA que se aproxime de forma arbitraria al comportamiento de cualquier función continua o continua a trozos (Liu, Yang y Cai, 2019).

Sim embargo, las características del MLP ponen limitaciones al conocimiento que tenemos sobre el funcionamiento de la red ya que el análisis teórico se vuelve bastante complicado por la existencia de una forma distribuida de no linealidad y la alta conectividad de la red (Haykin, 1999).

1.3. Redes convolucionales

Las redes convolucionales (CNN por sus siglas en inglés), son un tipo especializado de red que surge para procesar información que se sabe que tiene topología de rejilla, es decir, intenta reconocer patrones espaciales que sean invariantes ante traslaciones, escalamiento, rotaciones u otras distorsiones en arreglos de 1 o 2 dimensiones (Haykin, 1999; Nielsen, 2019). Esto las ha vuelto muy populares en la tarea de reconocimiento, clasificación y procesamiento de imágenes. La razón por la cual tienen dicho nombre es porque cuando menos una capa de la RNA utiliza la llamada operación de convolución.

Se puede considerar que la estructura de las CNN incluye los cuatro puntos siguientes (LeCun, Bengio & Hinton, 2015):

1. Conexiones locales: Para un MLP convencional se ordena toda la información en un arreglo unidimensional y se toma cada componente de una observación por separado. En una CNN se hacen conexiones en regiones localizadas, de tal forma que un campo receptivo local obliga a extraer características locales (Nielsen, 2019).
2. Compartición de pesos: En cada capa de la red se forman mapas de características, en forma de un plano, cuyas neuronas comparten pesos sinápticos. Esto a su vez nos otorga dos ventajas (Haykin, 1999):
   - Invarianza ante traslaciones.
   - Reducción del número de parámetros libres.
3. El pooling: Cada capa de convolución es seguida de una capa que realiza promedios locales y sub muestreo, tomando solo los valores más importantes y reduciendo el tamaño del mapa de características.
4. Múltiples capas: Este tipo de RNA tienen forma piramidal en la que la dimensión espacial de la imagen o del patrón se reduce conforme se avanza en la red, pero se aumenta el número de capas. Esto con el propósito de simplificar al máximo las características fundamentales del conjunto de entrada (Rojas, 1996).

En resumen, una red convolucional se define como un tipo de red neuronal profunda que emplea capas convolucionales con el propósito de identificar características clave de los datos de entrada y aprender sus patrones relevantes. Estas redes son altamente eficaces para el procesamiento de imágenes y señales, y han demostrado ser muy exitosas en una amplia variedad de aplicaciones.

1.3.1. Operación de convolución

La convolución es una operación realizada entre dos funciones reales a y b que, en cierto sentido, nos dicen como se superponen estas dos en el espacio

formando una nueva función f. En general se denota por (Goodfellow, Bengio y Courville, 2016):

$$f(u) = (a * b)(u) = \int a(v)b(u - v)dv = \int b(v)a(u - v)dv \qquad (13)$$

Aquí, *a(v)* se le suele conocer como la entrada, mientras que el segundo término *b* se conoce como el núcleo. Y la función que se obtiene *f(u)* es el mapa de características.

En el caso del análisis de una imagen bidimensional en una CNN se tiene que la entrada son las entradas de la neurona y el núcleo lo conforman los pesos que comparten todas las neuronas de la capa. De este modo, la convolución discreta realizada por la neurona k,l-ésima sobre un campo receptivo local de *n* por *m* esta dada por (Nielsen 2019):

$$f(u) = \sum_{i=0}^{n} \sum_{j=0}^{m} w_{i,j}\, a_{k+i,l+j} \qquad (14)$$

En este caso, tenemos un promedio ponderado de los datos en el campo receptivo local formando un filtro para los datos.

1.3.2. Arquitectura de las redes convolucionales

Una red convolucional consta de capas de neuronas artificiales organizadas en una matriz tridimensional con dimensiones correspondientes al ancho, alto y profundidad de los datos de entrada. La primera capa de la red lleva a cabo el proceso de convolución, de manera que se aplican filtros que forman mapas de características relevantes de la imagen. El resultado de la convolución, sumado a un cierto sesgo, se aplica a una función de activación no lineal, como el rectificador lineal unitario (ReLU), permitiendo capturar características de forma más compleja (LeCun, Bengio & Hinton, 2015).

A la capa de convolución le sigue una capa de pooling o submuestreo, utilizada para mezclar características similares, reduciendo la dimensión de los datos de salida. Esto se suele realizar tomando los máximos locales de unas secciones del mapa de características.

Es común tener una serie de capas de convolución seguidas de capas de submuestreo aumentando la cantidad de mapas de características, pero reduciendo la dimensión de los datos. Las redes convolucionales se suelen terminar con una capa o una red neuronal multicapa donde las neuronas se encuentran totalmente conectadas (LeCun, Bengio & Hinton, 2015).

1.3.3. Redes neuronales preentrenadas de Matlab

Matlab cuenta con una amplia variedad de redes neuronales profundas previamente entrenadas con algún subconjunto de imágenes de la base de datos ImageNet. Dicho subconjunto cuenta con más de un millón de imágenes de 1000 categorías diferentes (The MathWorks, Inc., 2024).

De entre las redes que se encuentran disponibles, solo dos arquitecturas tienen relevancia para la presente tesis:

1. Resnet: Las Redes Residuales (*Residual Networks* en ingles) son una familia de redes neuronales convolucionales desarrolladas por He, Zhang, Ren y Sun en 2015 para afrontar un problema con la precisión que se obtenía para las populares redes VGG (Visual Geometry Group) profundas.

   En general, cuando se aumenta el número de capas en las CNN's, aumenta la capacidad del modelo para representar funciones complejas. Pero el modelo tradicional VGG en el que se concatenan directamente las capas de convolución alcanza un valor crítico en su profundidad donde el error en la clasificación comienza a incrementar debido a que el gradiente usado como métrica para la retropropagación del error se va diluyendo conforme se distribuye hacia las capas iniciales. Esto resulta en porcentajes más bajos de precisión para redes más profundas (Boesch, 2023; He, Zhang, Ren y Sun, 2015).

Una forma de solventar este problema es a través de la implementación de un marco de aprendizaje residual profundo que se basa en el uso de atajos en conexiones de la red donde se utiliza la función identidad para normalizar el gradiente local como se observa en la Figura 2 (He et al., 2015).

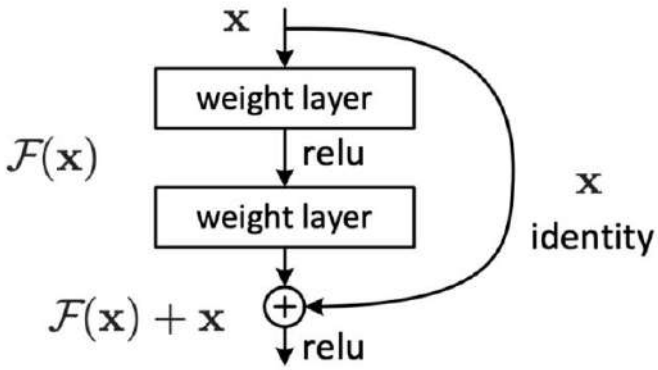

Figura 2. Esquema de atajo en conexión para el aprendizaje residual. (He, Zhang, Ren y Sun, 2015)

En este caso, Matlab cuenta con las Resnet18, Resnet50 y Resnet101, donde el número al final del nombre hace referencia a la cantidad de capas de convolución presentes en el camino más largo que conecta la capa de entrada con la capa de salida (The MathWorks, Inc., 2024; He et al., 2015), donde el tamaño de las imágenes de entrada es de 224 por 224 pixeles RGB.

2. Efficientnet: Son una familia de redes neuronales convolucionales desarrolladas por Tan y Le en 2019 tras realizar una investigación acerca de los métodos usados para escalar redes convolucionales para obtener mejores porcentajes de precisión. Comúnmente las CNN's se escalan al aumentar su profundidad o la resolución de las imágenes de forma arbitraria, siendo un proceso tardado y sin un patrón establecido.

   De forma intuitiva, si se aumenta la resolución de las imágenes, se requiere una mayor profundidad de red para incrementar el campo receptivo y un mayor ancho de red para que los nuevos canales capturen patrones detalles más finos. Por ese motivo, se desarrolló el modelo de

escalamiento compuesto, a través del cual se escala una red en 3 dimensiones de forma uniforme: ancho, profundidad y resolución de imagen; utilizando coeficientes compuestos que se pueden obtener a partir del modelo original (Tan y Le, 2019).

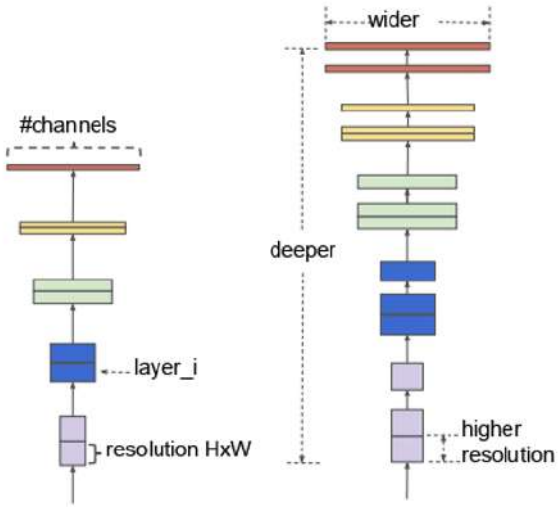

Figura 3. Esquema del escalamiento compuesto. Un ejemplo de red base (izquierda). Método de escalamiento compuesto para escalar las tres dimensiones con una tasa fija (Tan y Le, 2019)

Basados en esta nueva propuesta para escalar CNN's, Tan y Le desarrollaron una línea base de tamaño móvil llamada EfficientNet-B0 cuya arquitectura se base en bloques residuales de cuello de botella invertida de la MobileNetV2 (Sandler, M. et al., 2018). Esta red fue escalada con el método de escalamiento compuesto para dar lugar a la familia de redes EfficientNet, las cuales alcanzan porcentajes de precisión iguales a las otras redes convolucionales del momento, con la ventaja de ser más pequeñas y, en muchos casos, más veloces (Tan y Le, 2019).

En este caso, Matlab cuenta con la EfficientNetb0, que cuenta con 82 capas de convolución en el camino más largo que conecta la capa de entrada con la capa de salida (The MathWorks, Inc., 2024; Tan y Le, 2019), donde el tamaño de las imágenes de entrada es de 224 por 224 pixeles RGB.

Estas redes tienen el propósito de usarse para (The MathWorks, Inc., 2024):

- Clasificación de imágenes según las 1000 categorías del conjunto de entrenamiento previo.
- Extracción de mapas de características de un conjunto de imágenes para entrenar clasificadores como las máquinas de soporte vectorial, las cuales no requieren tantos recursos como el entrenamiento de una CNN completa.
- Entrenamiento parcial de la red para clasificar nuevos conjuntos de imágenes en un proceso conocido como transferencia de aprendizaje o *transfer learning*. Esto permite tomar las capacidades de extracción de características de la red como punto de partida para que el entrenamiento de dicha red para clasificar un nuevo conjunto de datos no parta desde cero (Tan et al., 2018).

Esta última aplicación es de especial relevancia ya que revela la posibilidad de tomar características de procesamiento útiles de las primeras capas de convolución de una red y aplicarlas a una nueva tarea de clasificación (Tan et al., 2018).

1.4. Reglas de optimización

Para el correcto funcionamiento de las RNA multicapa, cada observación de entrada tiene una salida esperada correspondiente, que determina el objetivo de salida, en otras palabras, lo que se espera que arroje la capa de salida según los datos que alimentan a la red. Sin embargo, el comportamiento específico que deban de tener el resto de las capas no está especificado directamente por los datos de entrenamiento (Goodfellow, Bengio & Courville, 2016). Por este motivo, el algoritmo de entrenamiento debe encontrar el modo de modificar eficientemente dichas capas para alcanzar el funcionamiento deseado de la RNA.

En la práctica, se usan métodos numéricos de optimización para minimizar el error de las salidas de una RNA, y técnicas como el gradiente conjugado, los algoritmos cuasi-Newtonianos, los de recocido o enfriamiento simulado y los

algoritmos genéticos han sido implementados (Cheng y Titterington, 1994). Sin embargo, los conceptos más importantes para el proyecto son los que se describen a continuación.

1.4.1. Descenso del gradiente

Diversas medidas para ajustar la RNA pueden ser utilizadas, pero es innegable que una de las más utilizadas es el error cuadrático medio (ECM) sobre las N entradas de entrenamiento y las K salidas (Navarro Pastor, 1998):

$$ECM(W) = \frac{1}{2NK}\sum_{n=1}^{N}\sum_{k=1}^{K}(y_{kn} - d_{kn})^2 \quad (15)$$

Donde y_{kn} es el valor predicho por la red y d_{kn} es el valor deseado para cada salida k de la red de cada entrada n.

Como ya sabemos, al entrenar una red neuronal modulamos los pesos sinápticos para alcanzar una configuración tal que permita hacer predicciones correctas, es decir, la función de coste. En este caso el error cuadrático medio, es una función de los pesos sinápticos de la RNA. Este error constituye una hipersuperficie en el espacio formado por los pesos sinápticos. El propósito del entrenamiento es hallar la configuración de W para que el ECM se minimice (Nielsen, 2019).

Existen diversos métodos para hallar el mínimo global, pero el más usado se basa en el hecho de que las derivadas de una función nos muestran la forma en la que estas cambian cuando existe una pequeña variación en sus parámetros. Entonces se puede encontrar una configuración W para la cual el error cuadrático medio sea menor al dar pequeños pasos en la dirección opuesta a la derivada del error (Bishop, 2006; Goodfellow, Bengio & Courville, 2016):

$$W^{(t+1)} = W^{(t)} - \epsilon \nabla E\left(W^{(t)}\right) \quad (16)$$

Donde *t+1* es la iteración siguiente a la *t*, y ϵ es un parámetro que define la tasa de aprendizaje, usualmente en el rango $0 < \epsilon < 1$.

1.4.2. Algoritmo de retropropagación del error

La configuración optima de pesos *W*, es un reto en el contexto de redes neuronales. Esto se debe a que en un arreglo multicapa, las capas ocultas no tienen asociado un error directamente como sucede en las capas de salida. La

solución de este problema supone la asignación de responsabilidades a cada nodo que forma la red sobre el resultado final para cada valor de la entrada (Gurney, 2004).

Para ello, se toma ventaja de que la función final que representa la red neuronal para transformar las entradas en las salidas puede pensarse como una concatenación de funciones que siguen la regla de la cadena del cálculo diferencial (Dinamarca, 2018).

En general, el algoritmo de retropopagación tiene los siguientes pasos (Nielsen, 2019):

1. Se introduce un dato de entrenamiento, el cual produce una cierta salida. Dicha salida tiene un error dado por la función de coste (como el ECM).
2. Se calcula el error asociado a la última capa δ_j^L como:

$$\delta_j^L = \frac{\partial E_c}{\partial y_j^L}\frac{\partial y_j^L}{\partial z_j^L} = \left(y_j^L - d_j\right) {f_j^L}'\left(z_j^L\right) \tag{17}$$

donde E_c es la función de error o función de costo, y_j^L es la salida de la j-ésima neurona de la capa L (la última), z_j^L es el campo local inducido de la j-ésima neurona de la capa L y ${f_j^L}'(u)$ es la derivada de la función de activación de la j-ésima neurona de la capa L.

3. Se retropropaga el error a la capa anterior δ_j^{l-1} con:

$$\delta_j^{l-1} = {f_j^{l-1}}'\left(z_j^{l-1}\right) \sum_{k=1} \delta_k^l w_{jk}^l \tag{18}$$

donde se consigue una expresión similar a la anterior pero donde los índices son $l = 1,2,\ldots,L$, y w_{jk}^l es el peso sináptico asociado entre la k-ésima neurona de la capa *l* y la neurona j-ésima de la *l-1*.

4. Se calcula las derivadas del error respecto a los pesos usando el error asociado:

$$\frac{\partial E_c}{\partial w_{ij}^l} = \delta_j^l y_i^{l-1} \tag{19}$$

Donde $l = 1,2,\dots,L$, los pesos de cada neurona van sobre $i = 0,1,2,\dots,n$, y *n* son cada uno de los enlaces que tiene esa neurona.

5. Se actualizan los pesos con la regla delta:

$$\Delta w_{ij} = -\epsilon \frac{\partial E_c}{\partial w_i} \tag{20}$$

1.4.3. Entropía cruzada

En la teoría de la información, la entropía es una especie de cuantificación de la incertidumbre asociada con una variable aleatoria, y la entropía cruzada es una comparación que se hace entre dos distribuciones de probabilidad P y Q para conocer la distancia que hay entre ellas (Rubinstein y Kroese, 2004; de Boer, Kroese, Mannor et al., 2005):

$$H(P,Q) = -\sum_i P_i log\ (Q_i) \tag{21}$$

Esta se minimiza cuando las dos distribuciones coinciden. En este sentido, la entropía cruzada se utiliza como una medida de la discrepancia que existe entre el valor real de salida, d_j, y el valor estimado por la red, y_j, lo cual, sumado sobre las *N* neuronas de salida proporciona una medida del error en la iteración (Kroese, Rubinstein y Taimre, 2006):

$$EC = \frac{-1}{N}\sum_i^N d_i * log\ (y_i) \tag{22}$$

Esta medida de la discrepancia que hay entre lo estimado y lo real puede minimizarse al emplear algoritmos como los que se presentan arriba. En general, la entropía cruzada es un estimador usado para tareas de clasificación debido a que la función logarítmica que contiene pondera en gran medida las estimaciones más alejadas de d_j y en menor medida las estimaciones más cercanas a los valores reales, lo que lleva a la convergencia del método (Margolin, 2004).

2. Haces estructurados de luz

2.1. Concepto

Se sabe que la luz tiene características tanto corpusculares como ondulatorias. En ese sentido, los haces de luz pueden ser descritos a través de una función $u(\vec{r},t)$ que cumple con la ecuación de onda (Condado Tepox, 2018):

$$\nabla^2 u(\vec{r},t) - \frac{1}{c_0^2}\frac{\partial^2}{\partial t} u(\vec{r},t) = 0 \qquad (23)$$

y describe su propagación en el espacio.

La radiación electromagnética transporta energía y momento. Toda interacción entre radiación y materia lleva a un intercambio de energía y, más importante en esta investigación, momento lineal o angular. En un principio se consideraba que la radiación electromagnética solo podría llevar momento lineal asociado a su propagación en el espacio. Sin embargo, autores como Allen, Padgett y Babiker (1999), y Yao y Padgett (2011) hacen referencia a los trabajos de Poynting (1909) y Beth (1936) por ser las primeras demostraciones teóricas y experimentales de la existencia del momento angular de espín (SAM), asociado a la polarización circular intrínseca de la radiación, al observar el torque que se induce sobre una placa birrefringente dependiendo de la helicidad del fotón que incide en ella, $\sigma = \pm 1$, asociada a la componente del SAM en la dirección de propagación dado por $\hbar\sigma$.

Más adelante, fue descubierto que la luz puede llevar momento angular orbital (OAM) que se relaciona con una polarización circular mayor que la que relacionada con el espín (Allen, Beijersbergen, Spreeuw & Woerdman, 1992; Allen, Padgett y Babiker, 1999).

El momento angular orbital suele estar inducido por una fase helicoidal dada por una dependencia de fase azimutal de la forma $e^{il\phi}$, donde l representa el número de momento angular orbital y ϕ es el ángulo azimutal (Lollie et al. 2022).

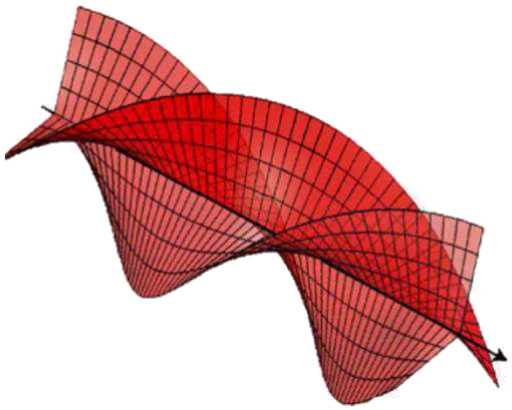

Figura 4. Frente de onda helocoidal (Yao y Padgett, 2011).

Usando las ecuaciones de Maxwell y las definiciones de densidad de momento lineal, angular y densidad de energía, se puede establecer que la relación entre el momento angular total y la energía total de los haces que exhiben esta disposición azimutal está dada por (Allen, Padgett y Babiker, 1999):

$$\frac{J_z}{W} = \frac{(l+\sigma)}{\omega} + \frac{\sigma}{\omega}\left[\frac{\int_0^k \frac{|E(\kappa)|^2\kappa}{(k^2-\kappa^2)}d\kappa}{\int_0^k \frac{|E(\kappa)|^2(2k^2-\kappa^2)}{\kappa(k^2-\kappa^2)}d\kappa}\right] \tag{24}$$

de donde se deduce que los momentos angulares orbital y de espín no pueden ser separados en la teoría clásica, debido al término adicional de corrección. Sin embargo, si se puede observar que, como el término adicional depende de σ, si la luz no posee polarización circular intrínseca, $\sigma = 0$, el haz aún puede poseer momento angular orbital. Dado que esta expresión solo proporciona la razón entre momento angular y energía, partiendo de la relación energía-frecuencia del fotón establecida por Einstein, $E = \hbar\omega$, el momento angular total también debe ir multiplicado por la constante de Planck reducida, de modo que un haz con estructura azimutal $e^{il\phi}$ cuenta con un OAM dado por $l\hbar$ por cada fotón (Allen, Beijersbergen, Spreeuw y Woerdman, 1992; Allen, Padgett y Babiker, 1999; Yao y Padgett, 2011).

2.2. Fundamentos teóricos

2.2.1. Ecuación de Helmholtz

Partiendo de las ecuaciones de Maxwell que describen a las interacciones electromagnéticas, junto con herramientas del cálculo vectorial, se puede llegar a que la luz obedece la ecuación (23), conocida como la ecuación de onda, cuyas soluciones se llaman funciones de onda, por ejemplo, puede tener como solución una función armónica en el tiempo dada por (Condado Tepox, 2018):

$$u(\vec{r},t) = A(\vec{r})cos\big(\omega t + \phi(\vec{r})\big) \tag{25}$$

donde $A(\vec{r})$ es la amplitud en función de la posición, el argumento del coseno es la fase de la onda, $\phi(\vec{r})$ es la fase inicial de la onda, ω es la frecuencia.

A través de la solución propuesta anteriormente se puede llegar a una solución de onda compleja (Páez López, 2017):

$$U(\vec{r},t) = A(\vec{r})e^{i\phi(\vec{r})}e^{i\omega t} = U(\vec{r})e^{i\omega t} \tag{26}$$

la cual, al sustituirse en la ecuación de onda, reduce la dependencia temporal a una constante multiplicada por la solución:

$$(\nabla^2 + \kappa^2)U(\vec{r}) = 0 \tag{27}$$

Conocida como la ecuación de Helmholtz, donde $\kappa = \frac{\omega}{c_0}$ es el número de onda.

Esta ecuación, en el caso general, puede tener múltiples soluciones dependiendo del sistema coordenado que se requiera o según varíen las condiciones de frontera (Páez López, 2017). Algunas de las soluciones físicamente realizables se obtienen al utilizar la aproximación paraxial, ya que las propiedades de propagación del haz en laseres reales permiten la conservación del perfil transversal sobre la dirección de propagación, siempre que el láser está razonablemente bien colimado (Allen, Padgett y Babiker, 1999). Algunas de ellas se presentan a continuación.

2.3. Haces gaussianos

Un haz gaussiano es un tipo de haz que no cuenta con una dependencia de fase azimutal, de modo que su distribución radial está dada por (Ruiz Ramírez, 2020):

$$u(r) = \sqrt{\frac{2P}{\pi w_0^2}} e^{-r^2/w_0^2} \tag{28}$$

donde r es la coordenada radial en coordenadas cilíndricas, P es la potencia del láser, w_0 es el ancho de la cintura del haz dentro del cual en 86.5% de la energía se concentra. Por otro lado, la cintura mínima del haz gaussiano ocurre en un punto focal después de haber pasado a través de una lente, después del cual el haz se expande y diverge. Se puede observar que su distribución de intensidad tiene simetría radial y se va reduciendo conforme nos alejamos del centro del haz.

2.4. Haces de Hermite-Gauss

Los haces de Hermite-Gauss son soluciones para la ecuación paraxial de Helmholtz en coordenadas rectangulares que tienen una distribución de intensidad dada por (Ruiz Ramírez, 2020):

$$u_{nm}^{HG}(x, y, z) = C_{nm}^{HG} \left[\frac{w_0}{w(z)}\right] H_n\left(\frac{\sqrt{2}x}{w(z)}\right) H_m\left(\frac{\sqrt{2}y}{w(z)}\right) e^{\frac{-x^2+y^2}{w^2(z)}} e^{-ikz} e^{ik\frac{x^2+y^2}{2R(z)}} e^{i(n+m+1)\zeta(z)} \tag{29}$$

donde C_{nm}^{HG} es la constante de normalización, $H_{n,m}$ son los polinomios de Hermite *n,m*-ésimos, y el radio del haz en la posición z es

$$w(z) = w(0)\sqrt{\frac{z_R^2 + z^2}{z_R^2}} \tag{30}$$

donde $w(0)$ es el ancho de la cintura del haz.

Los números cuánticos n y m determinan el orden del haz de Hermite-Gauss, donde el haz de orden $(n, m) = (0,0)$ es simplemente un haz gaussiano común. Estos haces no cuentan con ningún tipo de OAM asociado ya que no tienen dependencia azimutal (Ruiz Ramírez, 2020; Allen, Beijersbergen, Spreeuw y Woerdman, 1992). Aunque, Yao y Padgett (2011) señalan que Vaughan y Willets (1979) dedujeron que los modos de Hermite-Gauss de alto orden HG_{01} y HG_{10} pueden combinarse para obtener un haz con fase helicoidal.

2.5. Haces de Laguerre-Gauss

Un tipo especial de haz estudiado por sus propiedades de OAM y sus múltiples aplicaciones son los haces de Laguerre-Gauss (Páez López, 2017; Lollie et. al, 2015), los cuales tienen una distribución de amplitud dada por (Allen, Beijersbergen, Spreeuw y Woerdman, 1992; Allen, Padgett y Babiker, 1999):

$$u_{pl}^{LG} = \frac{C_{pl}^{LG}}{w(z)}\left(\frac{r\sqrt{2}}{w(z)}\right)^{|l|} e^{\frac{-r^2}{w^2(z)}} L_p^{|l|}\left(\frac{2r^2}{w^2(z)}\right) e^{\frac{-ikr^2 z}{2(z^2+z_R^2)}} e^{-il\phi} e^{i(2p+|l|+1)tan^{-1}\left(\frac{z}{z_R}\right)} \quad (31)$$

donde C_{pl}^{LG} es la constante de normalización, $L_p^l\left(2r^2/w^2(z)\right)$ es un polinomio de Laguerre generalizado y $(2p+l+1)tan^{-1}(z/z_R)$ es la fase de Gouy donde z_R es el rango de Rayleigh. Observando la dependencia azimutal se puede notar que los haces de Laguerre-Gauss tienen un OAM de $L = l\hbar$.

Yao y Padgett (2011) señalan que, desde antes del trabajo realizado por Allen, Beijersbergen, Spreeuw y Woerdman (1992), ya se conocía una propiedad importante de los haces de fase helicoidal: la singularidad de fase, que no es más que una línea totalmente oscura en el centro del haz. En la década de 1930, Dirac las encontró teóricamente y en la década de los 70, Nye, Berry y Wright encontraron experimentalmente las singularidades de fase en campos electromagnéticos.

Finalmente, ya que los haces de Laguerre-Gauss forman un conjunto de haces ortogonales, estos pueden ser empleados para incrementar la capacidad de transmisión en sistemas de comunicación óptica al superponerlos y propagarlos simultáneamente, en la llamada multiplexación de división de modo (MDM), que es un caso especial de la multiplexación de división de espacio (SDM). En este caso, la propiedad de ortogonalidad permite la demultiplexación eficiente y limpia de la información (Willner et al., 2015).

2.5.1. Dispersión en fibra óptica

En una fibra multimodo comercial cilíndrica de radio a, los modos están descritos por soluciones linealmente polarizadas (LP) que son soluciones de la ecuación de Helmholtz (Lollie et al., 2022):

$$LP_{lp} = N_{lp} \begin{cases} J_l(\kappa_{Tlp} r) e^{-il\phi} sir < a \\ K_l(\gamma_{lp} r) e^{-il\phi} sir \geq a \end{cases} \quad (32)$$

donde N_{lp} es una constante de normalización, $J_l(x)$ es la función de Bessel del primer tipo de orden l, mientras que $K_l(x)$ es la función de Bessel modificada del segundo tipo y de orden l. Los parámetros κ_{Tlp} y γ_{lp} dan el ritmo de oscilación del campo en el núcleo y en el revestimiento, respectivamente. Estos se definen por (Lollie et al., 2022):

$$\kappa_{Tlp}^2 = n_{nuc}^2 k_0^2 - \beta_{lp}^2, \gamma_{lp}^2 = \beta_{lp}^2 - n_{rev}^2 k_0^2 \quad (33)$$

donde $k_0 = 2\pi/\lambda_0$, con λ_0 siendo la longitud de onda en el vacío de la luz dentro de la fibra, β_{lp} es la constante de propagación del *p*-ésimo modo guiado para cada índice azimutal l, y n_{nuc} y n_{rev} son los índices de refracción del núcleo y el revestimiento, respectivamente. Para la descripción de los modos LP, se requiere el parámetro adicional de la fibra V, definido como (Lollie et al., 2022):

$$V^2 = \kappa_{Tlp}^2 + \gamma_{lp}^2 = \left(2\pi \frac{a}{\lambda_0}\right)^2 (n_{nuc}^2 - n_{rev}^2) \quad (34)$$

Este parámetro de la fibra determina la cantidad de modos y sus constantes de propagación. Esto implica que, sin importar la condición inicial, el modo de salida de la fibra siempre se puede escribir como una combinación lineal (Lollie et al., 2022):

$$\Psi_{salida}(r, \phi) = \sum_{l,p} c_{lp} \, LP_{lp} \quad (35)$$

donde los coeficientes c_{lp} están definidos por el campo injectado y las propiedades de la fibra óptica a través de la distancia de propagación. Una consideración clave para la luz espacial en fibras es la estructura helicoidal de un modo OAM, con su vórtice óptico sobre el eje de propagación. El producto de l con el índice azimutal ϕ da la carga topológica del modo, y el diámetro del

vórtice escala con números cuánticos l crecientes. Esto limita el diámetro azimutal de los modos acoplados dentro de la fibra. Si el diámetro del vórtice de los modos es mayor que el diámetro de la fibra, esto atenúa de forma severa la luz acoplada en la fibra (Wilner et al., 2015; Lollie et al., 2022).

En un escenario realista, las variaciones locales aleatorias de las propiedades de la fibra producen distorsiones significativas de los modos espaciales, haciendo casi imposible predecir la distribución espacial de fotones al final de la fibra, i.e., los coeficientes c_{lp}.

De forma similar las variaciones locales en las caracterísitcas del aire en el espacio libre también producen la dispersión de los modos OAM, provocando variaciones en su distribución espacial final.

III. HIPÓTESIS

Entrenando una red convolucional con las distribuciones de intensidad de luz de haces estructurados de tipo Laguerre-Gauss distorsionados por la propagación en un medio dispersivo es posible reconstruir el modo espacial correspondiente sin la necesidad de un procesamiento previo con al menos el 90% de eficiencia.

IV. OBJETIVOS

Objetivo general

Diseñar e implementar una red neuronal convolucional (CNN), con alto porcentaje de reconocimiento, para reconstruir modos espaciales de luz de tipo Laguerre-Gauss sometidos a distorsión por propagación en medios dispersivos, sin procesamiento previo de las distribuciones de intensidad de luz.

Objetivos específicos

I. Desarrollar un modelo teórico-computacional que permita generar las distribuciones de intensidad de modos espaciales orbitales Laguerre-Gauss. Con este modelo se buscarán las condiciones necesarias para la adquisición de imágenes en el montaje experimental del protocolo de comunicación usando modos espaciales, que se desarrollará en el Laboratorio de Micro- y Nano-fotónica del ICN-UNAM.

II. Crear una base de datos que integrará el conjunto de entrenamiento para el algoritmo de inteligencia artificial a partir de las imágenes adquiridas en el experimento antes mencionado.

III. Realizar pruebas con el conjunto de imágenes de las distribuciones de intensidad de luz variando los parámetros de la red neuronal convolucional para hallar la configuración plausible que permita obtener un porcentaje de reconocimiento de al menos 90%.

IV. Diseñar una interfaz gráfica de usuario amigable que permita recibir las distribuciones de intensidad de los modos espaciales orbitales distorsionados

por el medio dispersivo y reconstruya el modo espacial al que pertenece, sin la necesidad de un procesamiento previo.

V. Realizar pruebas finales de la red neuronal entrenada y comparación con otros modelos estadísticos dentro de la interfaz gráfica para corroborar el porcentaje de reconocimiento de al menos 90% y depurar el producto final con base en el tiempo de ejecución y la demanda de memoria.

V. METODOLOGÍA

1. Enfoque y nivel de la investigación

El enfoque de la investigación fue cuantitativo ya que en este se miden variables usando procesos estandarizados y se analizaron los datos usando métodos estadísticos, ambos con validez aceptada por la comunidad científica (Hernández Sampieri, Fernández Collado & Baptista Lucio, 2014) y, como ya fue establecido en la fundamentación teórica, una RNA puede considerarse una máquina que obtiene generalizaciones estadísticas de datos (Goodfellow, Bengio y Courville, 2016). Para el caso específico de esta investigación, la RNA nos permitió predecir, con un determinado grado de confianza, a qué modo espacial de Laguerre-Gauss pertenece una determinada distribución de intensidad de luz.

Se trabajó en el marco específico de la investigación experimental de campo el sentido que señala la Universidad Pedagógica Experimental Libertador (UPEL, 2016) “el análisis sistemático de problemas de la realidad con el propósito de (…) predecir su ocurrencia (…). Los datos de interés son recogidos en forma directa de la realidad; en este sentido se trata de investigaciones a partir de datos originales” (p. 18) ya que se obtuvieron las imágenes que conforman la base de datos para el entrenamiento de la CNN, según se establece en los objetivos específicos I y II.

Finalmente, la investigación tuvo un alcance exploratorio/correlacional (Hernández Sampieri, Fernández Collado & Baptista Lucio, 2014) ya que, a pesar del amplio estudio que se tiene sobre las propiedades de los haces que llevan OAM y su potencial para la comunicación óptica, es reciente la aplicación de RNA para su reconocimiento al propagarse en medios dispersivos (Lollie et al., 2022). La CNN entrenada identifica las correlaciones entre los patrones de intensidad de luz recibidos y los modos espaciales orbitales de entrada antes de la propagación.

2. Materiales

Para la presente investigación fueron necesarios los siguientes materiales:

2.1. Recursos tecnológicos

Como recursos tecnológicos se utilizó el software de cómputo numérico de la desarrolladora MathWorks®, Matlab en su versión R2023a (la más reciente hasta la fecha) disponible en la página sección de descargas de la página oficial:

https://www.mathworks.com/downloads/web_downloads

Se requiere la instalación las Toolbox de Deep Learning, Statistics and Machine Learning y App Designer.

Se utilizó una licencia académica otorgada por la Universidad Nacional Autónoma de México (UNAM).

Además, ya que la obtención de los patrones de intensidad de luz se llevó a cabo de forma independiente por el ICN-UNAM (institución externa), se cuenta con el permiso escrito y firmado por las autoridades competentes para el uso de las imágenes para los fines de investigación y desarrollo que se detallan en el presente documento.

2.2. Recursos materiales

Toda prueba y cálculo para la realización de los objetivos específicos I, III, IV y V se llevaron a cabo en una MacBook Pro con sistema operativo macOS Ventura versión 13.3.1 (a), con procesador de 2.3 GHz Intel Core i5 de dos núcleos y memoria RAM de 8 GB.

3. Metodología empleada

3.1. Modelo teórico computacional

Como ya se describió en la sección de antecedentes teóricos, los haces de luz que llevan momento angular orbital (OAM) suelen relacionarse con un término de fase $e^{il\phi}$, sin embargo, la distribución radial puede variar

dependiendo del marco en el que trabajemos (Allen, Padgett y Babiker, 1999). Para esta investigación se decidió utilizar los modos con OAM de tipo Laguerre-Gauss, los cuales tienen una distribución de amplitud dada por la ecuación (21).

En este sentido, se utilizó el lenguaje de programación Matlab para expresar estos modos orbitales usando las funciones matemáticas del programa y así poder simularlos, graficando la forma que tendrían las distribuciones de intensidad de los modos de Laguerre-Gauss al variar sus números cuánticos p y l. A modo de comparación, a continuación, se muestran dos de las simulaciones generadas en Matlab para las distribuciones de intensidad de los modos $LG_{0,1}$ y $LG_{0,7}$ respectivamente:

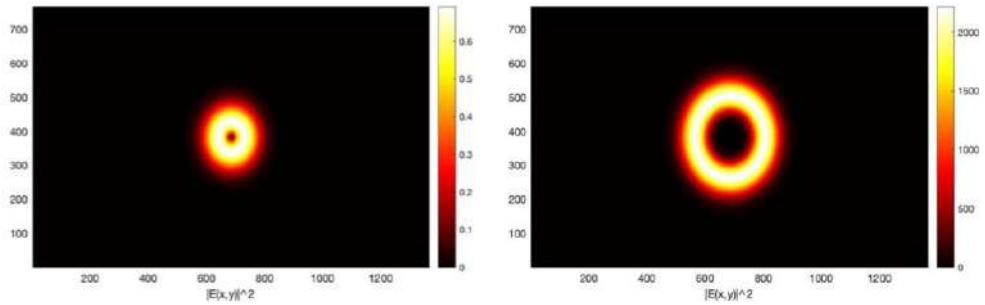

Figura 5. Simulación de las distribuciones de intensidad para los modos $LG_{0,1}$(izquierda) y $LG_{0,7}$(derecha)realizada en Matlab (Autoría propia).

Al estudiar las características de estos modos de Laguerre-Gauss, fue posible determinar de forma aproximada bajo qué condiciones se debe realizar el montaje experimental para implementar el protocolo de comunicación óptica a realizarse en el Laboratorio de Micro- y Nano-fotónica del ICN-UNAM. Aunque este montaje experimental se basa en gran medida en el que se describe en el trabajo de Lollie et al. (2022).

Hay que destacar que este proyecto de tesis no involucra el proceso de obtención de los patrones de intensidad de luz resultantes de la propagación de haces de Laguerre-Gauss en un medio dispersivo, si no que se limita a estudiar las condiciones para su obtención, la cual se llevó a cabo de forma independiente por el equipo del Laboratorio de Micro- y Nano-fotónica del

ICN-UNAM. Este es el motivo por el cual este documento no presenta la metodología para la obtención de los patrones de intensidad.

3.2. Desarrollo de la base de datos

Los patrones de intensidad de luz experimentales que se usan para entrenar a la CNN y fueron proporcionados por el Laboratorio de Micro- y Nano-fotónica del ICN-UNAM se administraron y se verificó que cuenten con características adecuadas para usarse por una RNA. Este proceso es una parte fundamental del proceso para implementar algoritmos de inteligencia artificial, tal y como lo relatan Provost y Fawcett (2013), en la mayoría de los casos, el éxito que puedan tener los algoritmos de ciencia de datos depende de la correcta definición de las variables a usar.

Para este experimento, se solicitaron 100 imágenes para cada uno de los modos de Laguerre-Gauss del tipo $LG_{0,l}$ con l tomando los valores enteros dentro del intervalo [-6,6] después de transmitirse por fibra óptica multimodo, y 100 imágenes para cada uno de los modos de Laguerre-Gauss del tipo $LG_{p,\pm 1}$ con p tomando los valores [0, 3, 5] luego de viajar por espacio libre. Esto nos da un total de 1,900 imágenes, que, aunque no sea un conjunto extremadamente grande, pueden tratarse usando las prácticas de la ciencia de datos para la preparación de información en el Big Data (Provost y Fawcett, 2013).

Cada imagen debe cumplir las siguientes características:

- Contar con un mismo formato, por ejemplo, todas deben ser de tipo (*.jpg).
- Contar con una etiqueta homogénea que permita la clara distinción de cada imagen. Esto se logró al etiquetar a cada una como “IMG_####”, donde cada uno de los símbolos de # representa un número. Así, por ejemplo, la primera imagen tuvo la etiqueta “IMG_0001” y la imagen número 1362 tuvo la etiqueta “IMG_1362”.
- Tener una resolución homogénea que sea relativamente baja, ya que, a mayor resolución, mayor cantidad de recursos serán necesarios para

implementar el algoritmo de inteligencia artificial. En este caso se utilizaron imágenes de 128 por 192 pixeles de un solo canal de color (escala de grises), e imágenes de 224 por 224 pixeles con tres canales de color (RGB). Esto se debe a que, para poder ingresar imágenes en alguna RNA se debe de preparar una capa de entrada que tenga las dimensiones de la imagen, y las redes implementadas en este proyecto contaban con capas de entrada de dimensiones [128, 192, 1] y [224, 224, 3], respectivamente.

Las imágenes se almacenaron en un disco duro con 19 carpetas, cada una para cada modo. Dichas carpetas fueron etiquetadas de forma tal que, de forma clara, nos dicen el modo al que corresponden: “m__#” para los que viajaron por fibra óptica y “LG_±1#” para los que viajaron por espacio libre. Es decir, las imágenes que corresponden a los modos $l = -1$ y $l = 9$ que viajaron por fibra óptica se encuentran dentro de las carpetas “m__-1” y “m__9”, respectivamente. Mientras que las imágenes que corresponden a los modos $l = -1,\ p = 5$ y $l = 1,\ p = 3$ que viajaron por espacio libre se almacenaron dentro de las carpetas “LG_-15” y “LG_13”, respectivamente. Esta es una característica importante del proceso que se aplica en la tarea de clasificación en RNAs, ya que así se puede saber cuál es la clase a la que pertenece cada imagen ingresada en la red y, así, entrenar la red o validar resultados.

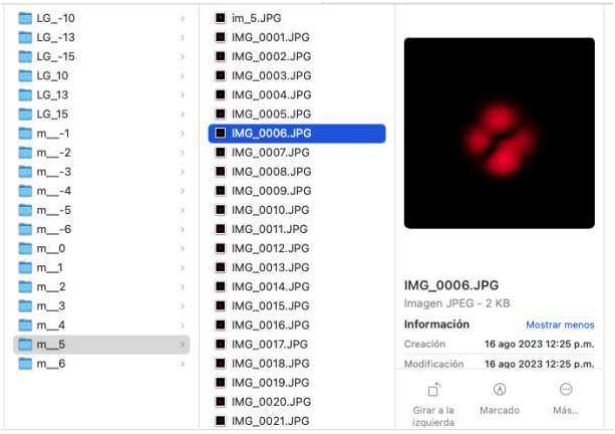

Figura 6. Almacenamiento de los patrones de intensidad de luz de 224 por 224 pixeles con 3 canales de color para el modo $LG_{0,5}$ (Autoría propia).

3.3. Entrenamiento y evaluación de la CNN

Para el desarrollo, entrenamiento, implementación en la GUI y evaluación de la CNN se siguió la siguiente metodología:

3.3.1. Redes para reconstrucción de modos espaciales

Se reprodujo el procesamiento de imágenes y la red multicapa usados en el trabajo de Lollie et al. (2022) para el reconocimiento de los modos orbitales de Laguerre-Gauss, con el objetivo de corroborar que es plausible su uso y para comparar resultados con la CNN en el último objetivo (Apéndice 1 y 2). Posteriormente, se programó una red convolucional con 3 capas de convolución, en la cual se coloca una capa de convolución, seguida de una de normalización, una unidad lineal rectificada (ReLU) para introducir la no linealidad y una capa de max pooling para reducir la dimensionalidad de los mapas de características (Figura 7). Esta red es la que tiene una capa de entrada de dimensiones [128, 192, 1] (para más detalles revisar el Apéndice 3).

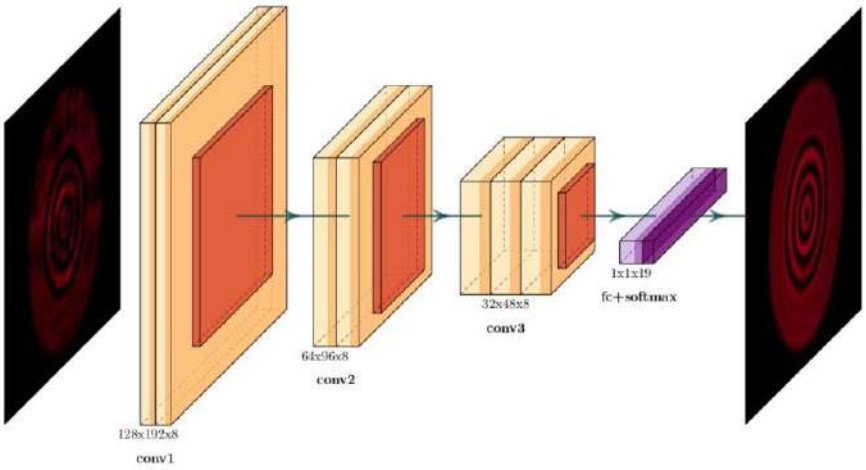

Figura 7. Arquitectura de la CNN simple de 3 capas de convolución para la reconstrucción de los modos LG (Autoría propia).

Además, se implementaron las redes preentrenadas "*efficientnet b0", "resnet 18", "resnet 50" y "resnet 101"* que se encuentran en las paqueterías de Matlab. Todas estas redes tienen capas de entrada de dimensiones [224, 224, 3] (para más detalles revisar los Apéndices 4 y 5).

Para realizar el entrenamiento de la primera red se varían los siguientes puntos:

- El parámetro ϵ (parámetro de entrenamiento) que controla la sensibilidad con la que se varían los pesos sinápticos en cada iteración del proceso de entrenamiento.
- El número de filtros de cada capa de convolución.

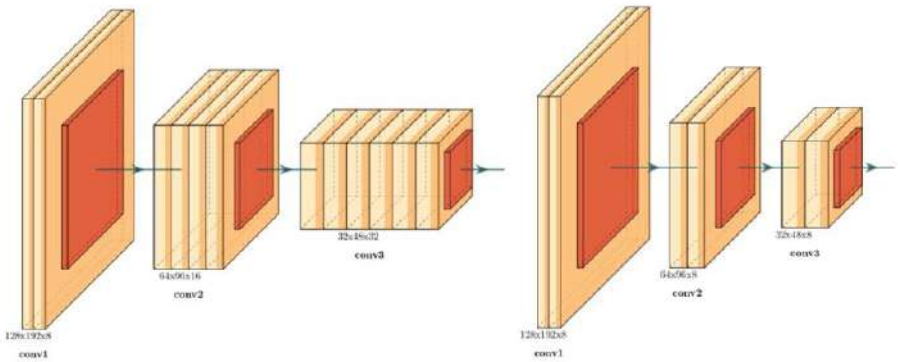

Figura 8. Diagrama de la CNN simple de 3 capas de convolución con 8, 18 y 32 filtros (izquierda) y 8, 8, y 8 filtros (derecha) (Autoría propia).

En este sentido se varía el ancho de la red con el propósito de cambiar la cantidad de mapas de características que se extraen de las imágenes a analizar.

- El tamaño de cada filtro que controla el campo receptivo local sobre el que se aplica la operación de convolución discreta de la ecuación (14), lo cual se relaciona de forma directa con la cantidad de pesos sinápticos de cada capa de convolución.

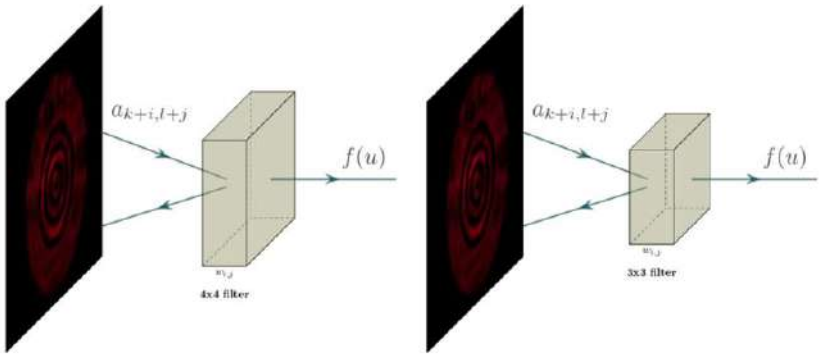

Figura 9. Representación de la operación de convolución con un filtro de 4 por 4 (izquierda) y 3 por 3 (derecha) (Autoría propia).

Esto con el objetivo de hallar una configuración de la CNN que requiera una menor cantidad de recursos (tiempo de ejecución y demanda de memoria) pero que siga obteniendo un porcentaje de reconocimiento mayor al 90%.
En las redes preentrenadas se varió la cantidad de capas congeladas, ya que, según se explica en la fundamentación teórica, se requiere entrenar de nuevo un cierto número de capas finales para que se adapten al conjunto de imágenes que se utiliza en este proyecto y pueda ocurrir la transferencia de aprendizaje para aprovechar la capacidad de las redes para clasificar imágenes (Tan et al., 2018).

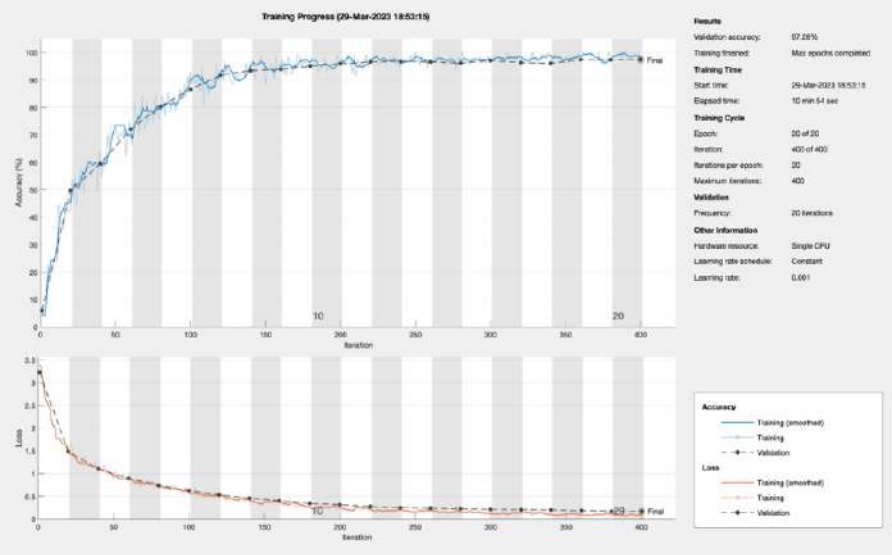

Figura 10. Gráficas y resultados generales de un entrenamiento de la CNN simple de 3 capas. Iteraciones contra precisión en azul (arriba) e iteraciones contra pérdidas en rojo (abajo) (Autoría propia).

La figura 10 muestra un ejemplo de las gráficas y resultados generales que tiene una de las ejecuciones de la CNN simple de 3 capas de convolución. Los resultados muestran que durante la etapa de entrenamiento la función de perdida decrece asintóticamente con el número de épocas, mientras que la precisión presenta un aumento asintótico, la cual alcanza 97.28% después de 400 iteraciones. Cabe mencionar que la precisión oscilada como una función de la épocas. Estas oscilaciones tienen amplitudes por debajo del 7%. En términos generales, esta arquitectura presenta desempeños plausibles que acuerdan con la complejidad del algoritmo y el tiempo de ejecución (Bishop, 2006; Goodfellow, Bengio & Courville, 2016; Nielsen, 2019).

Una vez se determinaron las redes con las configuraciones más plausibles, se utilizó la instrucción:

```
save('Nombre de la red', 'net');
```

Donde se le puede otorgar un nombre al archivo que contiene a la red entrenada para poder ser importada en la GUI y así usarse para hacer la clasificación de imágenes por parte del usuario.

3.4. Desarrollo de interfaz gráfica de usuario (GUI)

En esta sección se discute el desarrollo de la interfaz gráfica de usuario. Inicialmente, se delimitaron las funcionalidades que debe cumplir la GUI, así como las características generales, como:

- Implementación de un mecanismo para seleccionar la red con la cual se quiera realizar la reconstrucción del modo.
- Inclusión de un mecanismo para que el usuario seleccione el patrón a analizar, como un botón que abra el navegador de archivos del dispositivo.
- Uso de la imagen correspondiente al modo LG reconstruido por la red convolucional
- Inclusión de un texto que indique el modo al que corresponden el patrón original y el modo reconstruido, así como el medio en el que fueron propagados.

Y las características específicas con las que deberá contar para conseguir el objetivo de este trabajo:

- Diseño atractivo que otorgue información relevante para que el usuario sepa como interactuar con la interfaz de forma clara y concisa.
- Inclusión de logos de las instituciones involucradas en el desarrollo de la tesis.
- Métodos que eviten errores internos por las posibles interacciones de los usuarios.

- Inclusión del porcentaje de confianza que se tiene sobre la predicción realizada por la red convolucional.

Siempre tomando en consideración que la interfaz deberá ser sencilla de usar para usuarios que no estén familiarizados con la programación de redes neuronales (Microsoft, 2022).

Posteriormente se utilizó la toolbox App Designer de Matlab para realizar el diseño y programación de la GUI, ya que esta cuenta con herramientas visuales útiles conectadas con un código orientado a objetos que permiten una implementación más intuitiva de las funcionalidades necesarias para la interfaz gráfica.

Para la realización de la interfaz de usuario se revisaron los principios de la programación orientada a objetos para codificar las funcionalidades de los objetos que conforman a la interfaz, desde abrir la imagen de la distribución de intensidad de luz, hasta la obtención del modo angular reconstruido, pasando por el procesamiento de las imágenes realizado por la CNN previamente entrenada. Esto también incluye la programación necesaria para manejar las acciones que pueda realizar el usuario, como hacer clic sobre un botón para abrir el explorador de archivos, un menú desplegable para elegir la red con la cual quiera hacer la predicción y un botón para realizar la predicción y le permita al usuario conocer la confianza que se tiene sobre la predicción (Apéndice 6).

3.5. Prueba y depuración

El paso final es la realización de pruebas para verificar que todas las funcionalidades se lleven a cabo de forma correcta y para detectar cosas que haga falta incluir o que se puedan eliminar de la interfaz. En los múltiples casos en los que ocurrió un fallo, se procedió a depurar el código para corregir los errores de funcionamiento.

Adicionalmente, se buscaron formas de optimizar la interfaz y mejorar su rendimiento al variar la disposición de los componentes, alterar los códigos

para que sean más eficientes y cortos con el fin de reducir el tiempo de ejecución del programa y la memoria necesaria para su uso, y tomar en cuenta las acciones que podría realizar un usuario que pudieran conducir a un error en el código interno.

Se introdujeron diferentes patrones de entrada cuyos modos son conocidos para verificar que las salidas de la interfaz sean las esperadas. Para esto se tomaron en cuenta diversos índices de desempeño usados para los modelos de aprendizaje autónomo como lo son la especificidad, la sensibilidad y la precisión, los cuales se pueden hallar al construir la matriz de confusión (Google for Developers, s.f.).

VI. RESULTADOS Y DISCUSIÓN

Los resultados más relevantes para mostrar son los que corresponden a los puntos del 3.3 al 3.5 de la metodología.

1. Entrenamiento y selección de la red neuronal.

Como ya se mencionó con anterioridad, se implementó una red simple con 3 capas de convolución (Apéndice 3) y las redes pre entrenadas "*effincientnet b0*", "*resnet 18*", "*resnet 50*" y "*resnet 101*" que se encuentran en las paqueterías de Matlab (Apéndices 4 y 5).

Para ellas se variaron una serie de parámetros y se tomó en cuenta el tiempo de ejecución y la precisión alcanzada respecto al conjunto de prueba. En primer lugar, se exploró el desempeño de la red neuronal convolucional de 3 capas. Los resultados de las ejecuciones más significativas sen detallan a continuación:

Prueba No.	Tasa de aprendizaje (ε)	Tamaño del filtro	Numero de filtros			Tiempo de entrenamiento	Precisión (400 iters.)
			Capa 1	Capa 2	Capa 3		
1	1×10^{-3}	4	8	16	32	13 m 12 s	97.63%
2	1×10^{-4}	4	8	16	32	14 m 24 s	96.67%
3	1×10^{-2}	3	8	16	32	13 m 14 s	70.18%
4	1×10^{-3}	3	8	16	32	12 m 49 s	98.77%
5	1×10^{-4}	3	8	16	32	13 m 48 s	95.96%
6	1×10^{-3}	3	8	16	16	13 m 15 s	97.98%
7	1×10^{-3}	3	8	8	32	11 m 47 s	98.68%
8	1×10^{-3}	3	8	8	16	11 m 36 s	98.16%
9	1×10^{-3}	3	8	8	8	11 m 28 s	96.67%
10	1×10^{-3}	3	8	8	8	9 m 4 s	95.44%
11	1×10^{-3}	3	8	8	8	10 m 54 s	97.28%

Tabla 1. Resumen de pruebas y resultados de la CNN simple de 3 capas.

En este caso, se pude observar que en lo general esta arquitectura de red neuronal convolucional tiene resultados bastante satisfactorios en cuestión de tiempo de ejecución y precisión. Aunque, tomando en cuenta que se busca utilizar la menor cantidad de memoria posible, lo más adecuado sería escoger una arquitectura y parámetros como los que se muestran en las últimas tres pruebas de la tabla, ya que cuentan con la menor cantidad de filtros en total, manteniendo una precisión superior al 95%. Cabe destacar que esta elección no garantiza que la red más simple sea necesariamente la mejor para la tarea de reconocimiento de modos LG distorsionados, simplemente se busca obtener una red que optimice recursos computacionales (tiempo de ejecución y cantidad de memoria requerida) mientras mantiene un porcentaje de reconocimiento elevado (>90%), en acuerdo a lo que se planteó en la hipótesis.

Por otro lado, para la red pre entrenada "*efficientnet b0*" se obtuvieron los siguientes resultados:

Prueba No.	Tasa de aprendizaje (ε)	Capas congeladas (max 290)	No. de épocas	Tiempo de entrenamiento	Precisión (400 iters.)
1	1×10^{-3}	0	9	564 m 5 s	93.43%
2	1×10^{-3}	104	16	265 m 51 s	96.06%
3	1×10^{-3}	287	9	78 m 21 s	90.71%
4	1×10^{-3}	283	7	77 m 11 s	90.36%
5	1×10^{-3}	283	10	97 m 49 s	90.80%
6	1×10^{-3}	286	10	89 m 31 s	90.01%

Tabla 2. Resumen de pruebas y resultados de la CNN 'efficientnet b0' de Matlab.

Aquí se puede observar que se pueden conseguir buenos porcentajes de precisión, sin embargo, los tiempos de ejecución son muy largos y la red tiene un tamaño muy grande (290 capas), por lo cual esta red no es la mejor opción.

Finalmente, los tres tipos de redes "*resnet"* que se encuentran en la paquetería de Matlab fueron entrenadas obteniendo los siguientes resultados:

Prueba No.	Tasa de aprendizaje (ε)	Capas congeladas (max 347)	No. de épocas	Tiempo de entrenamiento	Precisión (380 iters.)
1	1×10^{-3}	342	10	175 m 14 s	93.87%
2	1×10^{-3}	332	8	151 m 16 s	95.71%
3	1×10^{-3}	322	10	206 m 2 s	96.41%

Tabla 3. Resumen de pruebas y resultados de la CNN 'resnet101' de Matlab.

Prueba No.	Tasa de aprendizaje (ε)	Capas congeladas (max 177)	No. de épocas	Tiempo de entrenamiento	Precisión (380 iters.)
1	1×10^{-3}	172	10	111 m 55 s	94.30%
2	1×10^{-3}	162	7	89 m 4 s	95.44%
3	1×10^{-3}	152	8	116 m 53 s	96.42%

Tabla 4. Resumen de pruebas y resultados de la CNN 'resnet50' de Matlab.

Prueba No.	Tasa de aprendizaje (ε)	Capas congeladas (max 72)	No. de épocas	Tiempo de entrenamiento	Precisión (380 iters.)
1	1×10^{-3}	66	10	43 m 1 s	91.41%
2	1×10^{-3}	59	10	46 m 40 s	95.71%
3	1×10^{-3}	50	7	39 m 55 s	97.11%
4	1×10^{-3}	43	10	68 m 37 s	96.93%

Tabla 5. Resumen de pruebas y resultados de la CNN 'resnet18' de Matlab.

donde se aprecia que todas estas arquitecturas regresan porcentajes altos de precisión, entre 91% y 97%. Sin embargo, se observa que el tiempo de ejecución en general es más grande para las redes que tiene más capas como la

"*resnet101"* y "*resnet50", con 347 y 177 capas respectivamente,* en comparación con la "*resnet18"* que cuenta con solo 72 capas.

Tomando los resultados anteriores en consideración, se decidió utilizar dos de estas redes:

- CNN simple de 3 capas, con ocho filtros por capa con un tamaño de tres por tres.
- "*Resnet18"* con 50 capas congeladas (que mantienen sus parámetros de preentrenamiento).

Ya que son las que tienen alto porcentaje de reconocimiento para los patrones, con una arquitectura de más baja dimensionalidad.

2. Desarrollo de la GUI.

En un primer intento por desarrollar una interfaz gráfica sencilla que permitiera llevar a cabo la selección del patrón de intensidad de luz y su posterior reconocimiento y clasificación según el modo LG predicho por la CNN, se decidió usar únicamente la red simple de 3 capas de convolución (Apéndice 3) con las imágenes de haces propagados en fibra óptica para obtener lo siguiente:

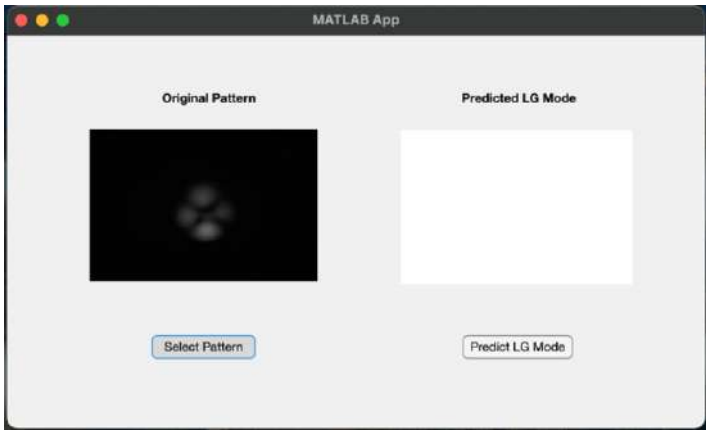

Figura 11. Primer diseño de la GUI luego de seleccionar un patrón de intensidad de luz para reconocimiento (Autoría propia).

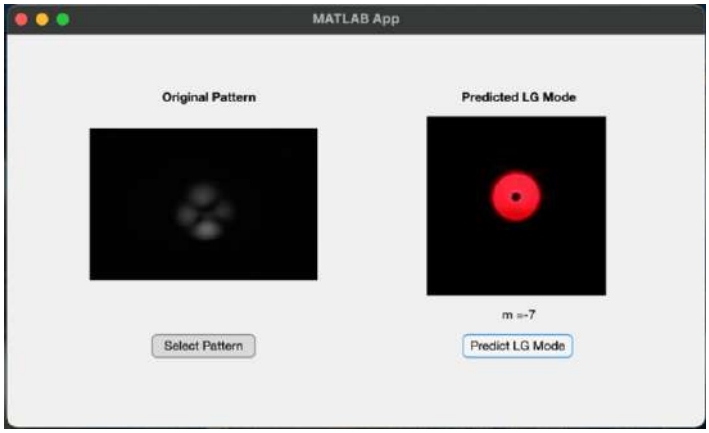

Figura 12. Primer diseño de la GUI luego oprimir el botón para realizar la predicción usando la red simple de 3 capas de convolución (Autoría propia).

La lógica que sigue este primer diseño de la GUI se desglosa en la Figura 13, pero se puede resumir en 3 partes:

1. Se inicializa la aplicación que contiene a la red simple de 3 capas de convolución y una imagen muestra de cada uno de los 13 modos LG antes de atravesar la fibra óptica.
2. Se permite al usuario seleccionar el patrón de intensidad del modo que desea reconocer al oprimir el botón “Select Pattern”.
3. Al oprimir el botón “Predict LG Mode” se llama a la red convolucional para llevar a cabo la clasificación. Este resultado se usa para llamar a la imagen muestra correspondiente y mostrar en texto el modo que se predijo.

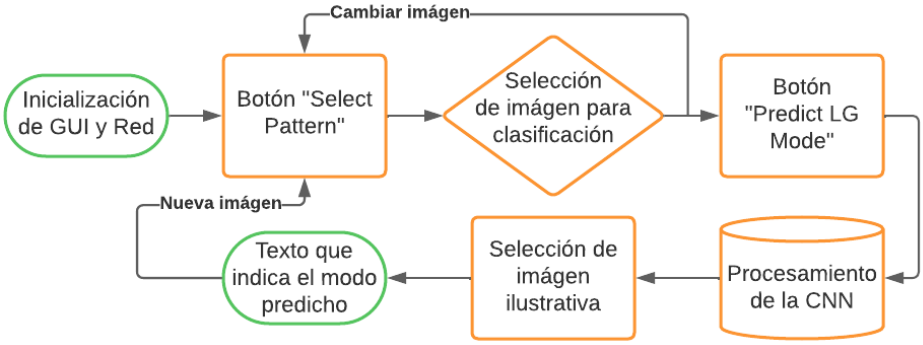

Figura 13. Diagrama de flujo de la lógica del código para el primer diseño de la GUI (Autoría propia).

Sin embargo, es claro que a este diseño requiere varias mejorías, iniciando por la inclusión de la red convolucional “*Resnet18*” con 50 capas congeladas (Apéndice 5), incluir imágenes de modos propagados en espacio libre, mostrar la comparación entre el modo real al que pertenece y el modo predicho, así como mostrar la confianza que se tiene en la predicción en cuestión.

Por este motivo se fueron realizando modificaciones de forma progresiva tanto al diseño como al código con el fin de que la GUI fuera más llamativa, amigable y mostrara información con mayor riqueza en detalles relevantes para la tarea. Así, finalmente, se llegó al siguiente diseño para la GUI:

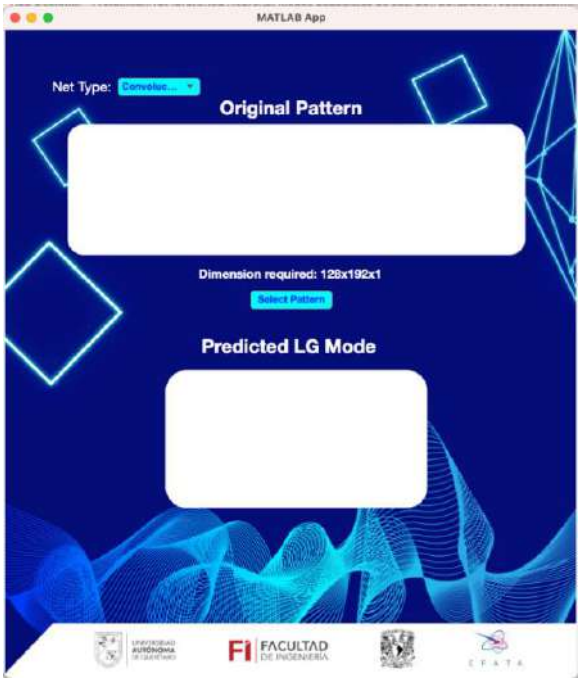

Figura 14. Diseño final de la GUI luego de ejecutar el código del Apéndice 6 (Autoría propia).

Aquí se pueden observar varios cambios, iniciando con dos nuevos fondos creados a la medida para la aplicación antes y después de iniciar el proceso de reconocimiento de los patrones de intensidad de luz.

En la parte superior izquierda se encuentra un menú desplegable que nos permite alternar entre la CNN simple con 3 capas de convolución y la "*Resnet18*" con 50 capas congeladas:

Figura 15. Menú desplegable que muestra las dos CNN que pueden ser usadas para el reconocimiento (izquierda) y GUI luego de seleccionar la "Resnet18" para realizar la clasificación (Autoría propia).

También se puede observar que para cada CNN se presenta un texto sobre el botón "*Select Pattern*" indicando la dimensionalidad de las imágenes que pueden ser procesadas por dicha red según se detalla en el último punto de la lista que se encuentra en la sección 3.2 de la Metodología.
En caso de que el usuario seleccionara una imagen con una dimensionalidad distinta a la que se indica, el programa no permite realizar el reconocimiento, simplemente resalta con rojo el texto que indica la dimensionalidad como se muestra en la figura 16.
Si se selecciona una imagen con las dimensiones adecuadas, la GUI obtiene automáticamente el modo LG al que pertenece el patrón al extraer el nombre de la carpeta en la que está contenida la imagen y se inicia una animación que simula como el modo LG original viaja por una fibra óptica (el medio dispersivo) y pierde su distribución espacial en el trayecto (Figura 17). Cabe destacar que esta parte de código solo funciona si las imágenes están almacenadas exactamente como se describe en la sección 3.2 de la Metodología, en caso de que se decida usar una nomenclatura o metodología diferente será necesario alterar el código del Apéndice 6.

Figura 16. GUI luego de seleccionar una imagen con dimensión distinta a la adecuada para la red seleccionada (Autoría propia).

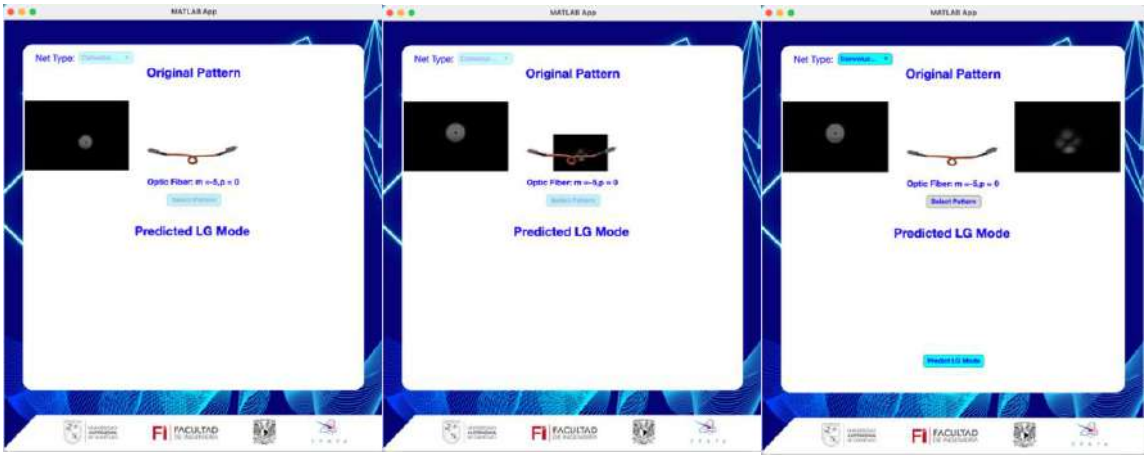

Figura 17. Sucesión de imágenes que muestran la animación que hace la GUI al escoger una imagen con la dimensión adecuada para la red seleccionada (Autoría propia).

Finalmente, aparece el botón "*Predict LG Mode*", el cual permite al usuario realizar la predicción del modo usando la CNN seleccionada. De forma similar al primer diseño de la GUI el programa muestra la imagen muestra que se tiene correspondiente al modo predicho y debajo de ella se encuentra el modo predicho y si corresponde a propagación en fibra óptica o en espacio libre.
Una mejora para obtener mayor información sobre la predicción es que también se muestra el porcentaje de confianza que se tiene en la predicción (Figura 18). Este porcentaje se obtiene al extraer todas las salidas de la última capa de la red, cada una de estas salidas corresponde a cada uno de los 19 modos

posibles. Sumando cada uno de los valores de las salidas se debe de obtener 1, de modo que estas salidas pueden interpretarse como probabilidades de que el patrón ingresado pertenezca a cada modo correspondiente.

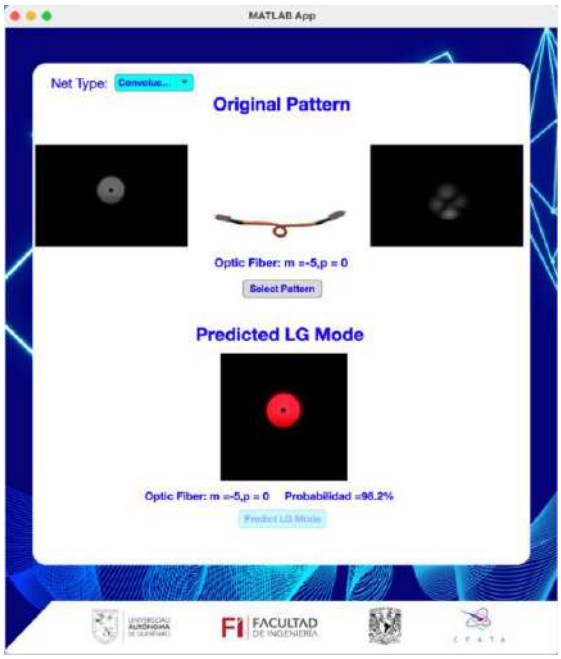

Figura 18. Diseño final de la GUI luego de oprimir el botón "Predict LG Mode" (Autoría propia).

En este caso la lógica de programación que sigue la GUI se desglosa en el diagrama de flujo de la Figura 19 y se puede resumir en:

1. Selección del tipo de red que se usará para la predicción.
2. Selección de la imagen a clasificar. Está condicionada a tener las dimensiones específicas según la red escogida.
3. Predicción del modo LG.

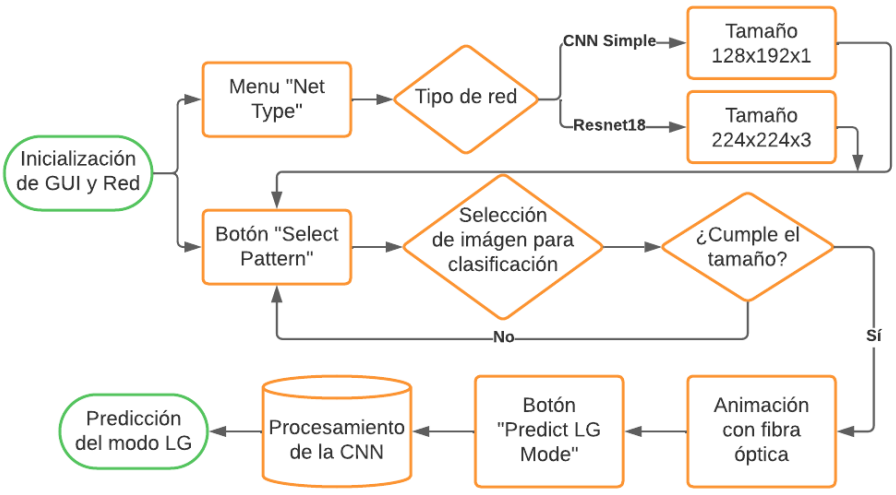

Figura 19. Diagrama de flujo de la lógica del código para el diseño final de la GUI (Autoría propia).

VII. CONCLUSIONES

En esta tesis se diseñó e implementó una CNN para la recontrucción de modos espaciales de luz de tipo Laguerre-Gauss distorsionados por propagación en medios dispersivos. En una primera etapa se desarrolló un modelo computacional para la simulación de las distribuciones de intensidad de los modos de Laguerre-Gauss. Estos resultados permitieron contrastar las distribuciones de intensidad obtenidas experimentalmente en el Laboratorio de Micro y Nano-fotónica del Instituto de Ciencias Nucleares.

A partir de las imágenes compartidas por el grupo de investigación del ICN se preparó una base de datos que habilitó el entrenamiento y validación de redes neuronales de convolucionales. La arquitectura de la red neuronal convolucional fue seleccionada a través de un estudio estadístico guiado por el desempeño del modelo en la tarea de clasificación. Con propósitos de comparación se reentrenaron parcialmente modelos como resnet y effincientnet. Las arquitecturas con los mejores resultados en términos de precisión y uso eficiente de recursos computacionales (menor espacio requerido y menor tiempo de ejecución) permitieron desarrollar y optimizar una GUI que facilitó el reconocimiento de los modos orbitales de luz distorsionados sin la necesidad de un procesamiento previo de las imágenes.

Tomando en cuenta los resultados que se muestran en las tablas 1 a 5 de la sección de Resultados, se puede apreciar que prácticamente todas las variantes de las redes convolucionales seleccionadas para el reconocimiento de las distribuciones de intensidad de luz de los haces estructurados de tipo Laguerre-Gauss distorsionados por la propagación en un medio dispersivo obtuvieron una precisión del 90% o superior, sin la necesidad de aplicar algún pre-procesamiento a dichas distribuciones de intensidad, demostrando que la hipótesis del presente proyecto de tesis es verdadera bajo las condiciones experimentales descritas en la metodología.

De entre las diferentes arquitecturas de CNN’s utilizadas para realizar el reconocimiento se destacan la CNN simple con 3 capas de convolución con 8

filtros de tamaño tres por tres en cada capa y la "*Resnet18*" con las primeras 50 capas congeladas, ya que estas presentaron altos porcentajes de reconocimientos con menores tiempos de ejecución y bajas dimensionalidades, lo cual representa un manejo eficiente de recursos computacionales sin sacrificar el alto porcentaje de reconocimiento de patrones (superior al 90%).

En conjunto, los resultados descritos en la sección anterior pueden tener impacto en los ámbitos científico y tecnológico:

I. Científico: El uso de RNA para la solución de problemáticas en diversas áreas de la ciencia ha ido en aumento gracias a su capacidad de extraer información relevante de conjuntos enormes de datos. En nuestro caso, la implementación de CNN's para la reconstrucción de patrones de intensidad provenientes de haces de Laguerre-Gauss que se propagaron en un medio dispersivo abre nuevas oportunidades para desarrollar metodologías de recuperación de información, codificación en haces estructurados para sistemas de comunicación óptica y la encriptación de información para generar canales seguros. Además, la implementación de la CNN en una interfaz gráfica tiene el potencial de reducir el tiempo usado en la parte computacional, dejando más espacio para la conceptualización de los fenómenos físicos involucrados o para el desarrollo de aplicaciones más complejas para estos haces de luz propagados en medios dispersivos.
II. Tecnológico: El dar este aporte a la investigación de los modos orbitales de luz, inevitablemente nos lleva al desarrollo e implementación de algoritmos para la comunicación óptica, pudiendo acercarse un poco más a usos que no sean meramente académicos, sino tecnológicos.

En futuras investigaciones o revisiones del material de la presente tesis, es relevante comentar las limitaciones que tienen estos resultados.

En primer lugar, este proyecto no se involucró con la experimentación para obtener las distribuciones de intensidad de los haces de Laguerre-Gauss distorsionados. Como ya se describió en la fundamentación teórica, el fenómeno

de dispersión que ocurre cuando los haces LG viajan por medios dispersivos es tan complejo matemáticamente que para montajes experimentales significativamente diferentes los patrones de intensidad de luz pueden cambiar de forma considerable, por lo cual es necesario estudiar la capacidad de las CNN's para alcanzar reconocimientos satisfactorios a pesar de estas variaciones.

En segundo lugar, solo se trabajó con haces LG propagados individualmente. En aplicaciones más realistas en el área de comunicaciones, una de las propiedades más interesantes de estos haces es su ortogonalidad, permitiendo aumentar la capacidad de transmisión de información (multiplexación de división de modo), por lo cual sería de especial importancia estudiar la capacidad de las CNN's para el reconocimiento de diversos modos LG propagados simultáneamente.

VIII. REFERENCIAS CONSULTADAS

[1] Allen, L., Beijersbergen, M. W., Spreeuw, R. J. C., & Woerdman, J. P. (1992). Orbital angular momentum of light and the transformation of Laguerre-Gaussian laser modes. Physical Review A, 45(11), 8185–8189. doi:10.1103/physreva.45.8185

[2] Allen, L., Padgett, M. J., & Babiker, M. (1999). IV The Orbital Angular Momentum of Light. Progress in Optics, 291–372. doi:10.1016/s0079-6638(08)70391-3

[3] Bhusal, N. et al (2021). Spatial Mode Correction of Single Photoms Using Machine Learning. Advanced Quantum Technologies. 4, 2000103. doi: 10.1002/qute.202000103

[4] Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Recuperado de https://www.microsoft.com/en-us/research/uploads/prod/2006/01/Bishop-Pattern-Recognition-and-Machine-Learning-2006.pdf

[5] Boesch, G. (2023). VGG Very Deep Convolutional Networks (VGGNet) – What you need to know. [Entrada de blog] Recuperado de https://viso.ai/deep-learning/vgg-very-deep-convolutional-networks/

[6] Bourennane, M., Karlsson, A., Bjork, G. et al. (2002). Quantum Key Distribution using Multilevel Encoding: Security Analysis. J. Phys. A: Math. Gen. 35 10065. doi: 10.1088/0305-4470/35/47/307

[7] Cañas, G. et al. (2017). High-dimensional decoy-state quantum key distribution over multicore telecommunication fibers. Phys. Rev. A 96, 022317. doi: 10.1103/PhysRevA.96.022317

[8] Cheng, B., & Titterington, D. M. (1994). Neural Networks: A Review from a Statistical Perspective. Statistical Science, 9(1), 2–30. doi:10.1214/ss/1177010638

[9] Condado Tepox, G. (2018). Estudio Numérico de la Propagación de Haces Pariaxiales. (Tesis de Licenciatura, Benemérita Universidad Autónoma de Puebla). Recuperado de https://repositorioinstitucional.buap.mx/handle/20.500.12371/7585

[10] Cozzolino, D. et al. (2019). Orbital Angular Momentum States Enabling Fiber-based High-dimensional Quantum Communication. Physical Review Applied, 11(6). doi:10.1103/physrevapplied.11.064058

[11] de Boer, PT., Kroese, D.P., Mannor, S. et al. (2005). A Tutorial on the Cross-Entropy Method. Ann Oper Res 134, 19–67. doi:10.1007/s10479-005-5724-z

[12] Dinamarca, A. (2018). Aprendizaje y Análisis de Redes Neuronales Artificiales Profundas. (Tesis de Licenciatura, Universidad Nacional de Cuyo, Argentina). Recuperado de https://bdigital.uncu.edu.ar/objetos_digitales/13989/dinamarca-agustina-tesina.pdf

[13] Ding, Y. et al. (2017). High-dimensional quantum key distribution based on multicore fiber using silicon photonic integrated circuits. npj Quantum Inf 3, 25. doi: 10.1038/s41534-017-0026-2

[14] Giordani, T. et al. (2020). Machine Learning-Based Classification of Vector Vortex Beams. Physical Review Letters, 124(16). doi:10.1103/physrevlett.124.160401

[15] Goodfellow, I., Bengio, Y. & Courville, A. (2016). Deep Learning. Recuperado de http://www.deeplearningbook.org

[16] Google for Developers. (s.f.). Evaluate Models Using Metrics. Recuperado de https://developers.google.com/machine-learning/testing-debugging/metrics/metrics?hl=en

[17] Gröblacher, S., Jennewein, T., Vaziri, A., Weihs, G. and Zeilinger, A. (2006). Experimental quantum cryptography with qutrits. New J. Phys. 8 75. doi: 10.1088/1367-2630/8/5/075

[18] Gurney, K. (2004). An introduction to neural networks. Recuperado de http://www.macs.hw.ac.uk/~yjc32/project/ref-NN/Gurney_et_al.pdf

[19] Haykin, S. (1999). Neural networks and learning machines. 3ra ed. Recuperado de https://cours.etsmtl.ca/sys843/REFS/Books/ebook_Haykin09.pdf

[20] He, K., Zhang, X., Ren, S. & Sun, J. (2015). Deep Residual Learning for Image Recognition. Microsoft Research. doi: 10.48550/arXiv.1512.03385

[21] Hofer, L. R., Jones, L. W., Goedert, J. L. & Dragone, R. V. (2019) Hermite–Gaussian mode detection via convolution neural networks. J. Opt. Soc. Am. A 36, 936-943. doi: 10.1364/JOSAA.36.000936

[22] Huang, Z. et al. (2021). All-Optical Signal Processing of Vortex Beams with Diffractive Deep Neural Networks. Physical Review Applied, 15(1). doi:10.1103/physrevapplied.15.014037

[23] Jurado-Navas, A., Tatarczak, A., Lu, X., Olmos, J. J. V., Garrido-Balsells, J. M., & Monroy, I. T. (2015). 850-nm hybrid fiber/free-space optical communications using orbital angular momentum modes. Optics Express, 23(26), 33721. doi:10.1364/oe.23.033721

[24] Kroese, D.P., Rubinstein, R.Y. & Taimre, T. (2007). Application of the cross-entropy method to clustering and vector quantization. J Glob Optim 37, 137–157. https://doi.org/10.1007/s10898-006-9041-0

[25] Le Cun, Y. (1986). Learning Process in an Asymmetric Threshold Network. Disordered Systems and Biological Organization, 233–240. doi:10.1007/978-3-642-82657-3_24

[26] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436–444. doi:10.1038/nature14539

[27] Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), 2278–2324. doi:10.1109/5.726791

[28] Little, W. A. (1974). The Existence of Persistent States in the Brain. From High-Temperature Superconductivity to Microminiature Refrigeration, 145–164. doi:10.1007/978-1-4613-0411-1_12

[29] Liu, Z., Yang, Y., & Cai, Q. (2019). Neural network as a function approximator and its application in solving differential equations. Applied Mathematics and Mechanics, 40(2), 237–248. doi:10.1007/s10483-019-2429-8

[30] Liu, Z., Yan, S., Liu, H., & Chen, X. (2019). Superhigh-Resolution Recognition of Optical Vortex Modes Assisted by a Deep-Learning Method. Physical Review Letters, 123(18). doi:10.1103/physrevlett.123.183902

[31] Lollie, M. L. J. et al. (2022). High-dimensional encryption in optical fibers using spatial modes of light and machine learning. Machine Learning: Science and Technology. 3 035006. doi: 10.1088/2632-2153/ac7f1b

[32] Malik, M. et al. (2012). Influence of atmospheric turbulence on optical communications using orbital angular momentum for encoding. Opt. Express 20, 13195-13200. doi: 10.1364/OE.20.013195

[33] Margolin, L. (2005). On the Convergence of the Cross-Entropy Method. Annals of Operations Research, 134(1), 201–214. doi:10.1007/s10479-005-5731-0

[34] McCulloch, W. S. & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. The Bulletin of Mathematical Biophysics, 5(4), 115–133. doi:10.1007/bf02478259

[35] Mhaskar, H. N. (1993). Approximation properties of a multilayered feedforward artificial neural network. Advances in Computational Mathematics, 1(1), 61–80. doi:10.1007/bf02070821

[36] Microsoft. (2022). Diseño de una interfaz de usuario. Recuperado de https://learn.microsoft.com/es-es/windows/win32/appuistart/designing-a-user-interface

[37] Mirhosseini, M. et al. (2015). High-dimensional quantum cryptography with twisted light. New J. Phys. 17 033033. doi: 10.1088/1367-2630/17/3/033033

[38] Navarro Pastor, J. B. (1998). Aplicación de las redes neuronales artificiales al tratamiento de datos incompletos. (Tesis doctoral, Universitat Autònoma de Barcelona). Recuperado de tdx.cat/bitstream/handle/10803/5488/TJBNP1de2.pdf

[39] Nielsen, M. (2019). Neural Networks and Deep Learning. Recuperado de http://neuralnetworksanddeeplearning.com/

[40] Páez López, R. (2017). Generación de haces estructurados para

manipulación de micropartículas. (Tesis doctoral, Instituto Nacional de Astrofísica, Óptica y Electrónica, México). Recuperado de https://inaoe.repositorioinstitucional.mx/jspui/bitstream/1009/358/1/PaezLoR.pdf

[41] Ponce Cruz, P. (2010). Inteligencia artificial con aplicaciones a la ingeniería. Ciudad de México: Alfaomega Grupo Editor. ISBN: 978-607-7854-83-8

[42] Provost, F. &Fawcett, T. (2013). Data Science for Business. Sebastopol: O'Reilly Media, Inc. ISBN: 978-1-449-36132-7

[43] Rojas, R. (1996). Neural Networks - A Systematic Introduction. Recuperado de http://personal.cimat.mx:8181/~amor/Academic/Books/Neural_Networks.pdf

[44] Rosenblatt, F. (1957). The perceptron: A perceiving and recognizing automaton. Recuperado de https://blogs.umass.edu/brain-wars/files/2016/03/rosenblatt-1957.pdf

[45] Ruiz Ramírez, A. M. (2020). Caracterización y Análisis de Operación de un Laser Titanio-zafiro de Amarre de Modos de Lente Kerr. (Tesis de maestría, Instituto Nacional de Astrofísica, Óptica y Electrónica, México). Recuperado de https://inaoe.repositorioinstitucional.mx/jspui/bitstream/1009/1976/1/RuizRAM.pdf

[46] Rubinstein, R. Y., & Kroese, D. P. (2004). The Cross-Entropy Method. Information Science and Statistics. doi:10.1007/978-1-4757-4321-0

[47] Rumelhart, D. E., Hinton, G. E. & McClelland, J. L. (1987). Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol 1: Foundations. Massachusetts: The MIT Press. ISBN: 9780262680530

[48] Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. Nature, 323(6088), 533–536. doi:10.1038/323533a0

[49] Sandler, M. et al. (2018). MobileNetV2: Inverted Residual and Linear Bottlenecks. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 4510-4520. doi: 10.48550/arXiv.1801.04381

[50] Sun, R., Guo, L., Cheng, M., Li, J., & Yan, X. (2019). Identifying orbital angular momentum modes in turbulence with high accuracy via machine learning. Journal of Optics. doi:10.1088/2040-8986/ab2586

[51] Tan, C. et al. (2018) A Survey on Deep Transfer Learning. Artificial Neural Networks and Machine Learning – ICANN 2018. ICANN 2018. Lecture Notes in Computer Science(), vol 11141. Springer, Cham. doi:10.1007/978-3-030-01424-7_27

[52] Tan, M. & Le, Q. V. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. International Conference on Machine Learning, 2019. doi: 10.48550/arXiv.1905.11946

[53] The MathWorks, Inc. (2024). Redes neuronales profundas preentrenadas. Recuperado de https://la.mathworks.com/help/deeplearning/ug/pretrained-convolutional-neural-networks.html

[54] Willner, A. E. et al. (2015). Optical communications using orbital angular momentum beams, Adv. Opt. Photon. 7, 66-106. doi: 10.1364/AOP.7.000066

[55] Yáñez Márquez, C., López Leyva, L. O. & Aldape Pérez, M. (2007). Neurona artificial de McCulloch & Pitts. Recuperado de http://repositoriodigital.ipn.mx/handle/123456789/8640

[56] Yao, A. M., & Padgett, M. J. (2011). Orbital angular momentum: origins, behavior and applications. Advances in Optics and Photonics, 3(2), 161. doi:10.1364/aop.3.000161

IX. APÉNDICES

1. Apéndice 1: Código de Matlab para la Red Neuronal Multicapa Simple

```
DownDiv=20;%Tamaño del cuadrado para el downsampling
class=19;%Número de clases en las que se clasifican las imágenes

%Leer las imagenes .jpg en los directorios imgPath de los modos de luz
images=[];
for i=-9:9
    imgPath ='/Volumes/INGFISICA/Red Neuronal Modos Espaciales de Luz/Fotos/m__';
    imgType='/*.jpg';
    images=[images; dir([imgPath num2str(i) imgType])];
end
[ren,col]=size(images);
Target=zeros(class,ren); %Se inicializa el target

for k=1:ren
    %Extraemos el número de la imágen y folder
    aux=str2double(regexp(images(k).name,'\d+[\.]?\d*','match'));
    C(k)=aux(end);
    aux2=regexp(images(k).folder,'\_','split');
    i=aux2{end};
    %Leer imagen
    NameImages=[imgPath,i,'\',images(k).name];
    I=imread(NameImages);
    Im = im2gray(I);%Escala de grises
    Icropped=imcrop(Im,[1900 1100 2899 2499]);%Recorta los bordes negros que no aportan
información
    Id=double(Icropped);%Cambio a valores numéricos para el downsampling
    I_down=DownSampling_v2(Id,DownDiv);%Función de Down Sampling en archivo adjunto
    [ren1,col1]=size(I_down);
```

```
    Data(:,k)=reshape(I_down,[ren1*col1,1]);%Guardar la imágen k como un solo vector de
entrada para la red
    Target(str2num(i)+10,k)=1;%Cada imagen corresponde a un modo espacial que va del -9 al
10, pero los targets se enumeran del 1 al 18
end

%% Neural network
Net = patternnet(5);
Net.divideFcn = 'dividerand';
Net.divideMode = 'sample';
Net.divideParam.trainRatio=0.7;
Net.divideParam.valRatio=0.15;
Net.divideParam.testRatio = 0.15;
NetTrained = train(Net,Data,Target);
genFunction(NetTrained,'NN_ideal','MatrixOnly','yes')
```

2. Apéndice 2: Función de DownSampling para el procesamiento previo de las imágenes

```
function [FieldD]=DownSampling_v2(FieldFinal,SquareSize)
[OutPixel,OutPixel2,x]=size(FieldFinal);
res=mod(OutPixel,SquareSize);
Limit=(OutPixel-res)/SquareSize;

res2=mod(OutPixel2,SquareSize);
Limit2=(OutPixel2-res2)/SquareSize;

   %Down sampling
   for m=1:Limit
      for p=1:Limit2
         ini_m=SquareSize*(m-1)+1;
         fin_m=SquareSize*(m);
         ini_n=SquareSize*(p-1)+1;
         fin_n=SquareSize*(p);
         Im=FieldFinal(ini_m:fin_m,ini_n:fin_n);
         FieldD(m,p)=sum(sum(Im))/(SquareSize^2);
      end
   end

end
```

3. Apéndice 3: Código de Matlab para la CNN de 3 capas

```
imgPath ='/Volumes/INGFISICA/Red Neuronal Modos Espaciales de Luz/Fotos 128x192';
imds = imageDatastore(imgPath,'IncludeSubfolders',true, ...
    'FileExtensions','.jpg','LabelSource','foldernames');
numTrainFiles = 0.7;
[imdsTrain,imdsValidation] = splitEachLabel(imds,numTrainFiles,'randomize');
img = readimage(imds,1);
[ren,col] = size(img);

layers = [
    imageInputLayer([ren col 1])

    convolution2dLayer(3,8,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(3,8,'Padding','same')
    batchNormalizationLayer
    reluLayer

    maxPooling2dLayer(2,'Stride',2)

    convolution2dLayer(3,8,'Padding','same')
    batchNormalizationLayer
    reluLayer

    fullyConnectedLayer(19)
    softmaxLayer
    classificationLayer];
```

```
options = trainingOptions('sgdm', ...
    'MaxEpochs',20,...
    'InitialLearnRate',1e-3, ...
    'Shuffle','every-epoch', ...
    'ValidationData',imdsValidation, ...
    'ValidationFrequency',20, ...
    'Verbose',false, ...
    'Plots','training-progress');

net = trainNetwork(imdsTrain,layers,options);

YPred = classify(net,imdsValidation);
YValidation = imdsValidation.Labels;

accuracy = sum(YPred == YValidation)/numel(YValidation)
save('CNN-8-8-8.mat', 'net');
```

4. Apéndice 4: Código de Matlab para la efficientnet b0

```
imgPath ='/Volumes/INGFISICA/Red Neuronal Modos Espaciales de Luz/Fotos 224x224x3';
imds = imageDatastore(imgPath,'IncludeSubfolders',true, ...
    'FileExtensions','.jpg','LabelSource','foldernames');
numTrainFiles = 0.7;
[imdsTrain,imdsValidation] = splitEachLabel(imds,numTrainFiles,'randomize');
[ren,col] = size(readimage(imds,1));
% Cargar la red preentrenada
net= efficientnetb0;
%Sustituir capas finales
lgraph = layerGraph(net);
[learnableLayer,classLayer] = findLayersToReplace(lgraph);
%Se cambia las capas finales para coincidir con el número de clases
numClasses = numel(categories(imdsTrain.Labels));
if isa(learnableLayer,'nnet.cnn.layer.FullyConnectedLayer')
    newLearnableLayer = fullyConnectedLayer(numClasses, ...
        'Name','new_fc', ...
        'WeightLearnRateFactor',10, ...
        'BiasLearnRateFactor',10);
elseif isa(learnableLayer,'nnet.cnn.layer.Convolution2DLayer')
    newLearnableLayer = convolution2dLayer(1,numClasses, ...
        'Name','new_conv', ...
        'WeightLearnRateFactor',10, ...
        'BiasLearnRateFactor',10);
end

lgraph = replaceLayer(lgraph,learnableLayer.Name,newLearnableLayer);
newClassLayer = classificationLayer('Name','new_classoutput');
lgraph = replaceLayer(lgraph,classLayer.Name,newClassLayer);
%% Congelar las capas iniciales
layers = lgraph.Layers;
```

```
connections = lgraph.Connections;
flayer=286;
layers(1:flayer) = freezeWeights(layers(1:flayer));
lgraph = createLgraphUsingConnections(layers,connections);
%% Establecer las opciones de entrenamiento
Minibatch=70;
Valfreq=floor(numel(imdsTrain.Files)/Minibatch);
options = trainingOptions('sgdm', ...
    'MiniBatchSize',Minibatch,'MaxEpochs', 10,...
    'InitialLearnRate',1e-3,'Shuffle','every-epoch', ...
    'ValidationData',imdsValidation,'ValidationFrequency',Valfreq, ...
    'Verbose',false, 'Plots','training-progress');
net = trainNetwork(imdsTrain,lgraph,options);

YPred = classify(net,imdsValidation);
YValidation = imdsValidation.Labels;

accuracy = mean(YPred == YValidation)
```

5. Apéndice 5: Código de Matlab para las redes resnet

```
imgPath ='/Volumes/INGFISICA/Red Neuronal Modos Espaciales de Luz/Fotos 224x224x3';
imds = imageDatastore(imgPath,'IncludeSubfolders',true, ...
    'FileExtensions','.jpg','LabelSource','foldernames');
numTrainFiles = 0.7;
[imdsTrain,imdsValidation] = splitEachLabel(imds,numTrainFiles,'randomize');
[ren,col] = size(readimage(imds,1));
% Cargar la red preentrenada
net= resnet18;
%Sustituir capas finales
lgraph = layerGraph(net);
[learnableLayer,classLayer] = findLayersToReplace(lgraph);
%Se cambia las capas finales para coincidir con el número de clases
numClasses = numel(categories(imdsTrain.Labels));
if isa(learnableLayer,'nnet.cnn.layer.FullyConnectedLayer')
    newLearnableLayer = fullyConnectedLayer(numClasses, ...
        'Name','new_fc', ...
        'WeightLearnRateFactor',10, ...
        'BiasLearnRateFactor',10);
elseif isa(learnableLayer,'nnet.cnn.layer.Convolution2DLayer')
    newLearnableLayer = convolution2dLayer(1,numClasses, ...
        'Name','new_conv', ...
        'WeightLearnRateFactor',10, ...
        'BiasLearnRateFactor',10);
end
lgraph = replaceLayer(lgraph,learnableLayer.Name,newLearnableLayer);
newClassLayer = classificationLayer('Name','new_classoutput');
lgraph = replaceLayer(lgraph,classLayer.Name,newClassLayer);
%% Congelar las capas iniciales
layers = lgraph.Layers;
connections = lgraph.Connections;
```

```
flayer=50;
layers(1:flayer) = freezeWeights(layers(1:flayer));
lgraph = createLgraphUsingConnections(layers,connections);
%% Establecer las opciones de entrenamiento
Minibatch=70;
Valfreq=floor(numel(imdsTrain.Files)/Minibatch);
options = trainingOptions('sgdm', ...
    'MiniBatchSize',Minibatch,'MaxEpochs', 10,...
    'InitialLearnRate',1e-3,'Shuffle','every-epoch', ...
    'ValidationData',imdsValidation,'ValidationFrequency',Valfreq, ...
    'Verbose',false, 'Plots','training-progress');
net = trainNetswork(imdsTrain,lgraph,options);

YPred = classify(net,imdsValidation);
YValidation = imdsValidation.Labels;
accuracy = mean(YPred == YValidation)
save('Resnet18-50FL.mat', 'net');
```

6. Apéndice 6: Código de Matlab App Designer para la GUI

```
classdef GUILaguerreGaussBeams < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure              matlab.ui.Figure
        NetTypeDropDown       matlab.ui.control.DropDown
        NetTypeDropDownLabel  matlab.ui.control.Label
        Label3                matlab.ui.control.Label
        Image                 matlab.ui.control.Image
        Image3                matlab.ui.control.Image
        FibraOptica           matlab.ui.control.Image
        Label2                matlab.ui.control.Label
        Label                 matlab.ui.control.Label
        PredictLGModeButton   matlab.ui.control.Button
        SelectPatternButton   matlab.ui.control.StateButton
        PredictedLGModeLabel  matlab.ui.control.Label
        OriginalPatternLabel  matlab.ui.control.Label
        Image2                matlab.ui.control.Image
        Fondo                 matlab.ui.control.Image
    end

    % Callbacks that handle component events
    methods (Access = private)

        % Code that executes after component creation
        function startupFcn(app)
            global net
            app.Fondo.ImageSource = imread('Fondo-1.jpg');
            app.Fondo.Visible = 'on';
            app.FibraOptica.Visible = 'off';
```

```
    app.Image2.Visible = 'off';
    app.Image.Visible = 'off';
    app.Image.Position = [76,515,550,128];
    app.Image3.Visible = 'off';
    app.Label.Text = "";
    TipoRed = app.NetTypeDropDown.Value;
    if strcmp(TipoRed, 'Convolucional Simple')
        net = load('CNN-8-8-8-Nuevo-9860.mat');
        app.Label2.Text = "Dimension required: 128x192x1";
    else
        net = load('Resnet18-50FL-Nuevo-9316.mat');
        app.Label2.Text = "Dimension required: 224x224x3";
    end
    app.Label2.FontWeight = 'bold';
    app.Label2.FontColor = 'w';
    app.Label2.FontSize = 15;
    app.Label3.Text = "";
    app.PredictLGModeButton.Visible = "off";
    app.PredictedLGModeLabel.FontColor = 'w';
    app.OriginalPatternLabel.FontColor = 'w';
    app.NetTypeDropDownLabel.FontColor = 'w';

end

% Callback function
function Image2Clicked(app, event)

end

% Value changed function: SelectPatternButton
function SelectPatternButtonValueChanged(app, event)
    global OriginalPattern1 u im
```

```
app.SelectPatternButton.BackgroundColor = 'g';
pause(0.1)
app.SelectPatternButton.BackgroundColor = 'c';
startupFcn(app)
[filename, pathname] = uigetfile({'*.jpg'},'File Selector');
fullpathname = strcat(pathname,filename);
OriginalPattern1 = imread(fullpathname);
tama = size(OriginalPattern1);
TipoRed = app.NetTypeDropDown.Value;
if strcmp(TipoRed, 'Convolucional Simple')
    dimension = [128 192];
else
    dimension = [224 224 3];
end
subdim = size(dimension);
if size(tama) == subdim
    if tama == dimension
        app.Fondo.ImageSource = imread('Fondo-2.jpg');
        app.Image.Visible = 'on';
        app.PredictedLGModeLabel.FontColor = 'b';
        app.OriginalPatternLabel.FontColor = 'b';
        app.NetTypeDropDownLabel.FontColor = 'b';
        app.Label2.FontColor = 'b';
        app.SelectPatternButton.Enable = "off";
        app.NetTypeDropDown.Enable = "off";
        modo = regexp(pathname,'\_','split');
        type = regexp(char(modo(end-1)),'\/','split');
        if size(char(type(end)),2) == 2
            modo = regexp(char(modo(end)),'\/','split');
            modo = char(modo(1));
            p = regexp(modo(end),'\d','match');
            if size(char(regexp(modo,'-\d','match')),1) == 1
```

```
        m = '-1';
    else
        m = '1';
    end
    app.Label2.Text = strcat('Free space: m = ', m, ',p = ', p);
    modoname = strcat('Ig_', modo,'.JPG');
else
    modo = regexp(char(modo(end)),'\/','split');
    modo = char(modo(1));
    app.Label2.Text = strcat('Optic Fiber: m = ', modo, ',p = 0');
    modoname = strcat('im_', modo,'.JPG');
end
OriginalPattern0 = imread(strcat(pathname, modoname));
if strcmp(TipoRed, 'Convolucional Simple')
    OrigPatt0 = cat(3, OriginalPattern0, OriginalPattern0, OriginalPattern0);
    OrigPatt1 = cat(3, OriginalPattern1, OriginalPattern1, OriginalPattern1);
else
    OrigPatt0 = OriginalPattern0;
    OrigPatt1 = OriginalPattern1;
end
app.Image.ImageSource = OrigPatt0;
pause(5)
app.Image3.ImageSource = OrigPatt0;
app.Image3.Position = [40 515 192 128];
app.Image3.Visible = 'on';
app.FibraOptica = uiimage(app.UIFigure);
app.FibraOptica.Position = [254,515,188,60];
app.FibraOptica.ScaleMethod = 'fit';
app.FibraOptica.ImageSource = 'fibra optica.png';
app.FibraOptica.Visible = 'on';
h = 250;
t = 0.01;
```

```
                for i = 1:h
                    de = 3*i/220;
                    app.Image.Position = [40+i 515 192/log2(2+ de) 128/log2(2+ de)];
                    pause(t)
                end
                app.Image.ImageSource = OrigPatt1;
                norm = log2(2 + 3*h/220);
                for i = 1:h
                    de = 3*i/220;
                    app.Image.Position = [45+h+i/1.5 515 192*log2(2+de)/norm
128*log2(2+de)/norm];
                    pause(t)
                end
                app.PredictLGModeButton.Visible = "on";
                app.PredictLGModeButton.Enable = "on";
            else
                app.Label2.FontColor = 'r';
                app.Label2.FontWeight = 'bold';
                app.Label2.FontSize = 17;
            end
        else
            app.Label2.FontColor = 'r';
            app.Label2.FontWeight = 'bold';
            app.Label2.FontSize = 17;
        end
        app.SelectPatternButton.Enable = "on";
        app.NetTypeDropDown.Enable = "on";
        app.SelectPatternButton.BackgroundColor = [0.96,0.96,0.96];
    end

    % Button pushed function: PredictLGModeButton
    function PredictLGModeButtonPushed(app, event)
```

```
global OriginalPattern1 net
app.PredictLGModeButton.BackgroundColor = 'g';
pause(0.1)
app.PredictLGModeButton.BackgroundColor = 'c';
LGMode = string(classify(net.net,OriginalPattern1));
TipoRed = app.NetTypeDropDown.Value;
if strcmp(TipoRed, 'Convolucional Simple')
   layer = "softmax";
else
   layer = "prob";
end
Probabilidad = activations(net.net,OriginalPattern1,layer,OutputAs="rows");
prob = fix(max(Probabilidad)*100*10^2)/10^2;
modo = regexp(LGMode,'\_','split');
if char(modo(1)) == "LG"
   p = regexp(modo(end),'\d','match');
   if size(char(regexp(modo(end),'-\d','match')),1) == 1
      texto = strcat(['Free space:' ...
         ' m = -1, p = '],p(end));
   else
      texto = strcat(['Free space: m = 1, p = '],p(end));
   end
   ImageDir = strcat("LG_",modo(end),".png");
else
   texto = strcat(['Optic Fiber: m = '],modo(end), ',p = 0');
   ImageDir = strcat("im_",modo(end),".JPG");
end
app.Label.Text = texto;
app.Label3.Text = strcat('Probabilidad = ', string(prob), '%');
PredictedLGMMode = imread(ImageDir);
app.Image2.Visible = 'on';
app.Image2.ImageSource = PredictedLGMMode;
```

```
        app.PredictLGModeButton.Enable = "off";
    end

    % Value changed function: NetTypeDropDown
    function NetTypeDropDownValueChanged(app, event)
        startupFcn(app);
    end
end
% Component initialization
methods (Access = private)



% App creation and deletion
methods (Access = public)

    % Construct app
    function app = GUILaguerreGaussBeams

        % Create UIFigure and components
        createComponents(app)

        % Register the app with App Designer
        registerApp(app, app.UIFigure)

        % Execute the startup function
        runStartupFcn(app, @startupFcn)

        if nargout == 0
            clear app
        end
    end
```

```
        % Code that executes before app deletion
        function delete(app)

            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```