Universidad Autónoma de Baja California
Instituto de Ingeniería
Maestría y Doctorado en Ciencias e Ingeniería

Clasificación de imágenes SEM a partir de técnicas de aprendizaje profundo

Tesis para Obtener el Grado de:
Maestro en Ciencias

Presenta:

Luis Alberto Quintero López

Director de Tesis:

Dr. Jesús Caro Gutiérrez

Codirector de Tesis:

Dr. Félix Fernando González Navarro

Mexicali, B. C. Enero del 2025

Dedicatorias

A mis padres, por su apoyo incondicional a lo largo de toda mi formación académica.

A mis hermanas, por estar presentes en todos mis logros, paso a paso.

A mis profesores, por haberme guiado en este proceso académico.

A mis compañeros y amigos, por formar parte de mi vida y compartir estos logros.

Agradecimientos

Al Consejo Nacional de Humanidades, Ciencias y Tecnologías (CONAHCyT), por el apoyo económico brindado como estudiante de tiempo completo.

Al Dr. Jesús Caro Gutiérrez, mi director de tesis, gracias por apoyarme a lo largo de mi trayectoria formativa en el campo del aprendizaje profundo y por alentar el desarrollo de esta tesis.

Al Dr. Félix Fernando González Navarro, por compartir sus conocimientos en las diferentes materias dentro del área y por sus valiosos consejos.

A mi comité académico: la Dra. Brenda Leticia Flores Ríos, el Dr. Gabriel Alejandro López Morteo y el Dr. Jorge Eduardo Ibarra Esquer, quienes generosamente dedicaron su tiempo a la revisión de los avances de este trabajo.

A la Universidad Autónoma de Baja California (UABC), gracias por darme la oportunidad de formar parte de su comunidad estudiantil.

A mis padres, hermanas y al resto de mi familia, gracias por su apoyo incondicional y por alentarme a seguir adelante.

Resumen

En la actualidad, el desarrollo de nuevas tecnologías está fuertemente relacionado al estudio de nanomateriales. La nanotecnología es un campo de investigación prometedor que implica la manipulación de nanomateriales y cuenta con una gran variedad de aplicaciones. Para estudiar estos nanomateriales, los investigadores utilizan microscopios electrónicos de barrido, los cuales permiten generar y almacenar imágenes. Los diferentes usuarios encargados de etiquetar estas imágenes tienden a hacerlo de acuerdo con criterios subjetivos, lo que genera una gran cantidad de datos difíciles de gestionar. Ante esta problemática, el presente trabajo tiene como objetivo desarrollar una arquitectura de aprendizaje profundo que permita clasificar automáticamente las imágenes de nanomateriales, facilitando así su gestión. Para ello, se emplearon cuatro arquitecturas de aprendizaje profundo: CUSTOM, AlexNet, VGG16 y ResNet50. Se realizó una comparación entre ellas para determinar cuál ofrece un mejor desempeño en la tarea de clasificación, basándose en las métricas de exactitud, precisión, sensibilidad y puntaje-F1. Los resultados obtenidos muestran que los modelos CUSTOM y AlexNet presentaron un rendimiento inferior en la tarea de clasificación, con exactitudes promedio del 63.88% y 82.08%, respectivamente. En cambio, los modelos VGG16 y ResNet50 destacaron por obtener los mejores resultados, alcanzando exactitudes promedio del 95.41%. Aunque no se observó una diferencia significativa entre VGG16 y ResNet50, se recomienda optar por el modelo VGG16, debido a que posee un número de capas considerablemente menor que ResNet50, lo que se traduce en un menor consumo de recursos computacionales. Esta ventaja hace que VGG16 sea más rápido en términos de tiempo de entrenamiento y

lo convierte en una opción más viable, especialmente en entornos con limitaciones de poder computacional.

Contenido

Figuras

Tablas

1. Introducción

Los nanomateriales se clasifican en función del número de dimensiones que tienen. Pueden agruparse en cuatro categorías: dimensión cero, unidimensionales, bidimensionales y tridimensionales. Cada grupo tiene sus propias propiedades físicas, forma o tamaño. Los nanomateriales son materiales que tienen al menos una dimensión dentro del rango de la nanoescala, que es inferior a 100 nm [1]. Pueden obtenerse de forma natural o intencionada por medios físicos o químicos [2]. Hoy en día, el desarrollo de nuevas tecnologías depende en gran medida del estudio de los nanomateriales, como las nanopartículas (NPs, por sus siglas en inglés) , los nanotubos (NTs) y los nanopicos (NSs) como se muestra en Fig. 1.1. Las NPs pueden sintetizarse a partir de diversos metales, como oro, hierro, platino y óxidos metálicos. Siendo las NPs de plata de las más utilizadas y cuyas propiedades físicas y conductoras son bien conocidas [3]; suelen verse en las imágenes de forma circular y presentan patrones relativamente sencillos. Los NTs son diminutas estructuras cilíndricas tubulares o alineadas verticalmente. Algunos aparecen desordenados y pueden encontrarse en materiales como el níquel, el cobalto, el oro o una combinación de estos metales. Los NTs de carbono son utilizados habitualmente por sus propiedades únicas, como la conductancia eléctrica y la resistividad [4]. Los NSs tienen forma puntiaguda y su aspecto puede ser similar al de otros tipos de nanomateriales en ciertos ángulos, cada uno con su propia forma única. Los NSs se obtienen de infusiones en todo tipo de materiales algunos como: carbono, oro, diamante dopado con boro [5] y titanio [6]. Estos suelen ser utilizados en el área dental [7]. Debido a sus propiedades únicas, las NPs, NTs y NSs tienen una amplia gama de aplicaciones en el campo de los antibióticos [8] y las propiedades antibacterianas [9]. A nanoescala los NTs de carbono, también pueden utilizarse para generar sistemas fotovoltaicos de alta eficiencia capaces de convertir la energía solar en electricidad a bajo coste [10] y los NSs mejoran el rendimiento de las células solares gracias a su alta capacidad de recolección de carga y su rápida transferencia de electrones [11]. El Laboratorio de Semiconductores, Microelectrónica y Nanotecnología del Instituto de Ingeniería de

la Universidad Autónoma de Baja California, se enfoca en la caracterización del metal-oxido-semiconductor (MOS, por sus siglas) y su aplicación en dispositivos electrónicos, así como en estructuras de arseniuro de galio y silicio. Cuando los nanomateriales presentan patrones distintos, pueden distinguirse fácilmente a simple vista en las imágenes. Uno de los instrumentos más utilizados para estudiar a estos nanomateriales es el microscopio electrónico de barrido (SEM, por sus siglas en inglés), este es un dispositivo versátil utilizado habitualmente en nanociencia y nanotecnología para explorar la estructura de muestras bajo una resolución de 1 nm [12]. El SEM utiliza un haz de luz para crear una imagen detallada de la superficie de un objeto. Dispone de varios detectores, entre ellos el detector de electrones secundarios que permite la generación de imágenes de alta resolución, un detector de electrones retrodispersados que permite obtener imágenes de la composición y topografía de la superficie, y un detector de energía dispersiva que recoge los rayos X generados por la muestra, lo que permite realizar análisis semicuantitativos y de distribución.

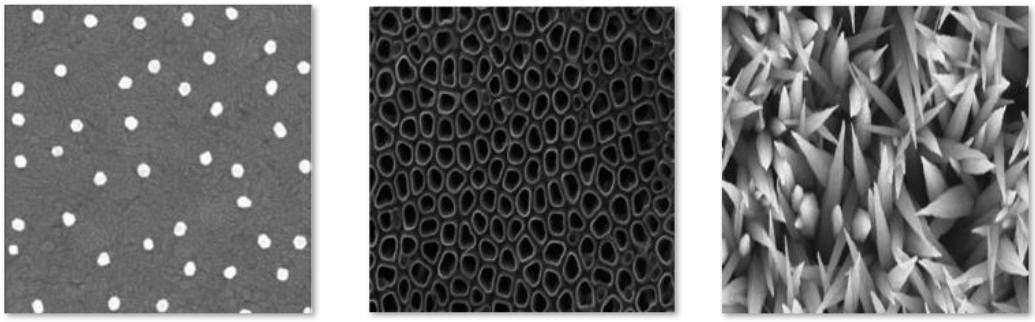
1.1 Planteamiento del problema

En la mayoría de los centros o institutos de investigación que disponen de un SEM, hay varios laboratorios con sus áreas de investigación y sus muestras de estudio. Estos laboratorios tienen sus usuarios encargados de utilizar el SEM para generar imágenes de la estructura de sus muestras. Como resultado, se genera un gran número de imágenes por usuario y área de investigación.

Actualmente, todas las imágenes del SEM se almacenan en unidades centrales de procesamiento y se etiquetan según los criterios subjetivos de los usuarios como se observa Fig. 1.1, lo que limita enormemente su explotación y repetibilidad a largo plazo, debido a la falta de un paradigma de clasificación bien definido. Uno de los principales retos de la investigación científica, incluida la nanociencia, es la gestión a largo plazo de los datos producidos. La Fig. 1.1(a) muestran NPs, mientras que la Fig.1.1(b) muestra NTs y por otra parte la Fig.1.1(c) muestra NSs, estas revelan las imágenes mal etiquetadas por los usuarios (Fig.1.1). Mostrando que es difícil para los inexpertos clasificar los etiquetados mostrados.

A la hora de crear una base de datos para un experimento, uno de los mayores obstáculos es la adquisición de las imágenes adecuadas. Muchas imágenes no están disponibles para su uso, y las que sí lo están no suelen estar organizadas adecuadamente. Como resultado, un experto debe revisar y clasificar las imágenes manualmente en sus diferentes categorías, lo cual suele consumir demasiado tiempo y ser una tarea laboriosa. Además, el etiquetado de las imágenes a menudo no proporciona información suficiente sobre sus características, lo que dificulta su reconocimiento y uso por parte de los usuarios. Esta falta de datos relevantes en

las etiquetas puede generar incertidumbre y confusión al intentar interpretar las imágenes correctamente, lo que hace aún más compleja la tarea de clasificación y análisis.



(a) NPs (b) NTs (c) NSs

Figura 1.1: Etiquetado subjetivo de los usuarios. Como se puede observar, el etiquetado de las imágenes se realiza de manera subjetiva y varía según el usuario. En la muestra (a), el etiquetado comienza con la magnificación de la muestra, seguida del tipo de materiales. En la muestra (b), el etiquetado inicia con el número de la muestra, seguido de la magnificación, tipo de electrones con los que se genera la imagen y la velocidad de escaneo. En la muestra (c), el etiquetado comienza con la magnificación de la muestra, seguida de algunas letras cuyo significado es desconocido. Esta falta de un un estándar en los criterios de etiquetado contribuye significativamente a la dificultad de gestionar y organizar los datos de manera eficiente.

Este problema ha llevado a los investigadores a buscar métodos complementarios y alternativos en técnicas como el aprendizaje profundo, que permiten mejorar la gestión de los datos producidos a través de la clasificación de imágenes, así como etiquetado de diversos nanomateriales.

1.2 Justificación

Este trabajo presenta modelos de aprendizaje profundo para clasificar automáticamente imágenes de diferentes tipos de nanomateriales, como NPs, NTs y NSs, lo que representa un avance significativo en la nanotecnología. Sin embargo, el verdadero desafío radica en el reconocimiento de patrones dentro de las imágenes, ya que las imágenes de diferentes clases pueden compartir patrones similares, lo que genera confusión. Aunque las redes neuronales profundas (DNN, por sus

siglas en inglés) han demostrado ser eficaces en tareas complejas de clasificación, siempre existe un grado de incertidumbre asociado a estos modelos, especialmente cuando las imágenes contienen características complejas o de baja calidad. Este problema se acentúa cuando las imágenes presentan detalles sutiles o patrones no representados en el conjunto de entrenamiento [13], lo que puede generar errores en la clasificación.

A pesar de los avances en la disponibilidad de recursos de hardware y redes preentrenadas, la clasificación automática mediante aprendizaje profundo no está exenta de limitaciones. Los modelos pueden enfrentar dificultades al procesar datos no presentes durante el entrenamiento, lo que resalta la importancia de abordar las posibles fuentes de error para mejorar la precisión de los resultados. Existen modelos como los generativos que podrían complementar la clasificación, proporcionando más datos y mejorando la capacidad de generalización, aunque esto incrementaría la complejidad y el coste computacional.

En este contexto, las DNN se aplican para abordar problemas como la detección de anomalías en imágenes obtenidas con el SEM [14], y son capaces de aprender representaciones jerárquicas y realizar etiquetado [15]. Sin embargo, a pesar de los avances prometedores en la clasificación automática, la supervisión humana sigue siendo un componente esencial para garantizar la precisión y confiabilidad de los resultados. La combinación de técnicas automatizadas y validación humana representa la estrategia más efectiva para reducir errores y asegurar que el sistema sea eficiente, preciso y confiable en la clasificación de imágenes [16].

La evolución de la clasificación de imágenes a partir de categorías predefinidas ha permitido un análisis más detallado y preciso de las imágenes, facilitando la identificación y clasificación de diferentes nanomateriales y patrones. Este avance es fundamental para aplicaciones avanzadas en nanotecnología y materiales, donde la automatización del proceso de clasificación juega un papel crucial para mejorar la eficiencia y precisión en el manejo de grandes volúmenes de datos visuales.

1.3 Objetivos

1.3.1 Objetivo general

Desarrollar una arquitectura para la clasificación de imágenes de nanomateriales basada en técnicas de aprendizaje profundo.

1.3.2 Objetivos específicos

- Realizar una revisión de la literatura sobre clasificación de imágenes de nanomateriales usando aprendizaje profundo.
- Crear un conjunto de datos con imágenes de tres tipos de nanomateriales y su etiquetado correspondiente.
- Preprocesar el conjunto de datos para su uso en modelos de aprendizaje profundo.
- Implementar y comparar modelos de redes neuronales convolucionales para la clasificación de imágenes, evaluando su rendimiento y costos computacionales.
- Evaluar que la clasificación de imágenes sea aceptable usando un conjunto de imágenes de referencia.

2. Marco teórico

En este capítulo se presentan los conceptos fundamentales necesarios para llevar a cabo una investigación sobre la clasificación de imágenes SEM utilizando técnicas de aprendizaje profundo. El objetivo es proporcionar una comprensión clara de la dirección que toma esta investigación, así como de la metodología empleada para lograr su meta.

2.1 Nanomateriales

Los nanomateriales son materiales que contienen partículas de una o más dimensiones a nanoescala, es decir, desde aproximadamente 1 nanómetro hasta 100 nanómetros. El nanómetro equivale a una milmillonésima parte de un metro (1 nm $= 10^{-9}$ m). Los nanomateriales se presentan de forma natural, por ejemplo, en las cenizas de los volcanes, o elaborados intencionalmente, creados a partir de algún proceso industrial, como los humos de soldadura o los productos de combustión [17].

Al modificar el tamaño de los nanomateriales, sus propiedades pueden cambiar, como un aumento en la conductividad eléctrica, variaciones en características físicas como el diámetro o la altura, mayor resistencia, propiedades magnéticas, así como una mejor interacción con otros materiales.

Gracias a su versatilidad, la nanotecnología es el área encargada del entendimiento de los nanomateriales y de transmitir aquellos resultados no entendibles, con el objetivo de crear herramientas tecnológicas para la humanidad en ámbitos como la ingeniería, las telecomunicaciones, la medicina, la farmacéutica e incluso en el sector eléctrico [18].

2.1.1 Clasificación de nanomateriales

Existen procesos que permiten clasificar los nanomateriales según su forma, tamaño y composición, considerando tanto sus estructuras internas como externas. Existen nanoestructuras de origen natural, como lo suelen ser las biológicas (por ejemplo, los virus y bacterias), o del medio ambiente, como la niebla y el humo causados por incendios forestales.

Algunos materiales tienen su origen en actividades involuntarias relacionadas con la naturaleza humana, como los productos de combustión. Otros, en cambio, provienen de actividades voluntarias, como la fabricación de nanomateriales manufacturados. Estos materiales suelen ser diseñados con propiedades específicas y se encuentran dentro del ámbito de los nano-objetos, los cuales pueden distinguirse por tener una, dos o incluso tres dimensiones a escala nanométrica.

El proceso de creación de los nano-objetos presenta estructuras en nanoescala. Entre ellas están algunas como las NPs, los NTs y los NSs [19]. Durante este proceso, los nano-objetos se descomponen, y algunos de estos materiales tienden a expandirse, lo que hace que alcancen tamaños superiores a 100 nm. En el caso de partículas débiles que no alcanzaron dicho tamaño, se suman a la formación de superficies pertenecientes a componentes individuales, mientras que las partículas fuertemente unidas suelen caracterizarse por tener un núcleo o matriz sólida y permanecen intactas.

Los materiales con estructura interna o superficial a nanoescala se consideran nanoestructurados, como los nanopicos. Estos materiales pueden adoptar la forma de polvo nanoestructurado, nanocompuesto, nanoespuma sólida, materiales nanoporosos y nanodispersión fluida. En la Fig. 2.1 se presenta un esquema de cómo se dan dichos nanomateriales.

2.2 Microscopio electrónico de barrido

El SEM es un instrumento crucial para la observación de estructuras a escala nanométrica, utilizado principalmente en el análisis de nanomateriales como nanopartículas, nanotubos y nanopicos. Gracias a su alta resolución y capacidad de aumento, el SEM permite obtener imágenes detalladas que facilitan el estudio topográfico, estructural y composicional de las muestras. La calidad de las imágenes generadas depende de parámetros como el tamaño del punto, el ángulo de apertura y la intensidad del haz de electrones, los cuales influyen directamente en la precisión de los modelos de clasificación de imágenes aplicados a los nanomateriales. En el ámbito de la clasificación de imágenes, el SEM desempeña un papel clave, ya que las imágenes de alta resolución obtenidas permiten identificar y clasificar con mayor exactitud las diferentes estructuras

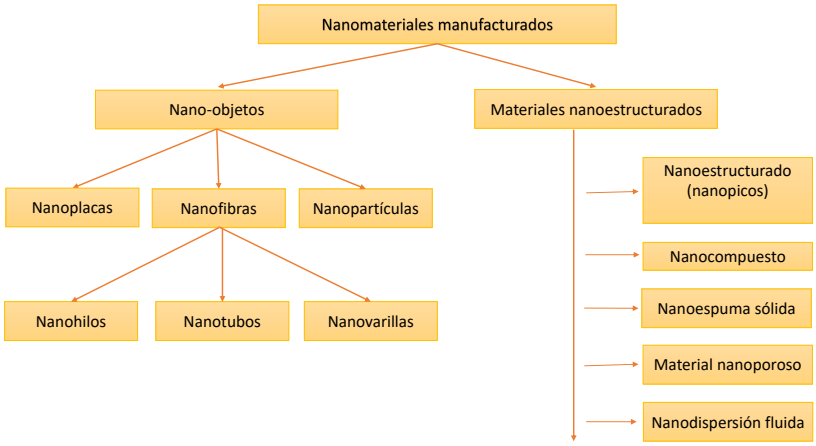


Figura 2.1: Esquema representativo de los nanomateriales manufacturados.

de los nanomateriales. Los electrones secundarios (SE, por sus siglas en inglés) y retrodispersados (BSE, por sus siglas en inglés), generados por la interacción del haz de electrones con la muestra, proporcionan información crítica sobre la morfología y la composición atómica de los materiales. Los SE son más abundantes y útiles para obtener datos morfológicos detallados, mientras que los BSE permiten obtener información sobre la distribución atómica y los elementos presentes en la muestra. Las partes principales del SEM se pueden ver en la Fig. 2.2. Para lograr una clasificación precisa de imágenes, es fundamental que el SEM proporcione imágenes de alta calidad, lo que requiere ajustar adecuadamente parámetros como el tamaño del punto y la intensidad del haz. Estos ajustes optimizan la claridad y el detalle de las imágenes. Lo anterior es crucial para alimentar algoritmos de DL que puedan identificar patrones específicos en las imágenes y clasificar correctamente los diferentes tipos de nanomateriales.

El SEM opera con tres etapas de vacío, diseñadas para asegurar que el haz de electrones se mantenga dirigido correctamente hacia la muestra. Estas etapas, organizadas de arriba hacia abajo según la estructura del microscopio (ver Fig. 2.2), cumplen funciones específicas para el control del haz. Primera etapa: Es la zona donde se alcanza el vacío más alto, ya que es aquí donde se generan los electrones. Un vacío de alta calidad es esencial para evitar que los electrones pierdan energía o se desvíen. Segunda etapa: Está ubicada en la parte central del microscopio, donde se encuentran las bobinas de exploración. El vacío es ligeramente menor que en la etapa anterior. Esta sección tiene como función ajustar y controlar el haz de electrones antes de que llegue a la muestra, asegurando así una focalización precisa y estable. Tercera etapa: Está ubicada cerca de la

muestra; el vacío es el más escaso. El vacío en esta zona puede variar, permitiendo trabajar con vacíos bajos, medios o altos, según los requiera la muestra.

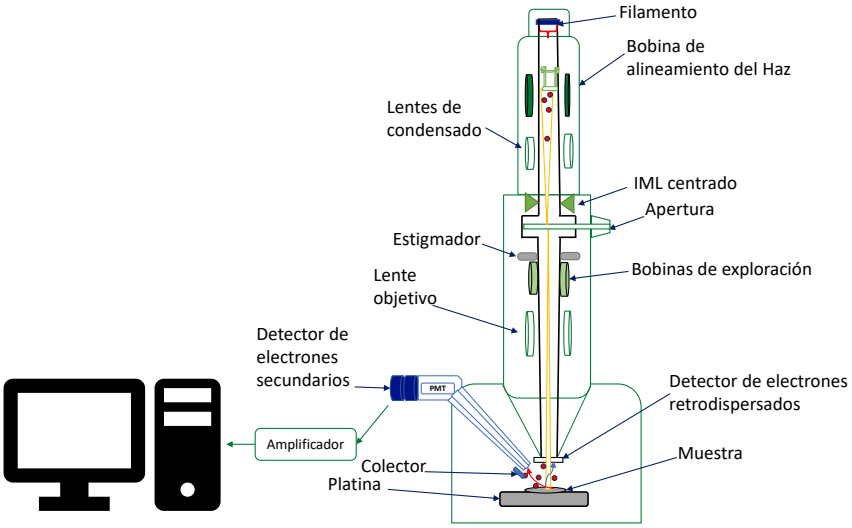
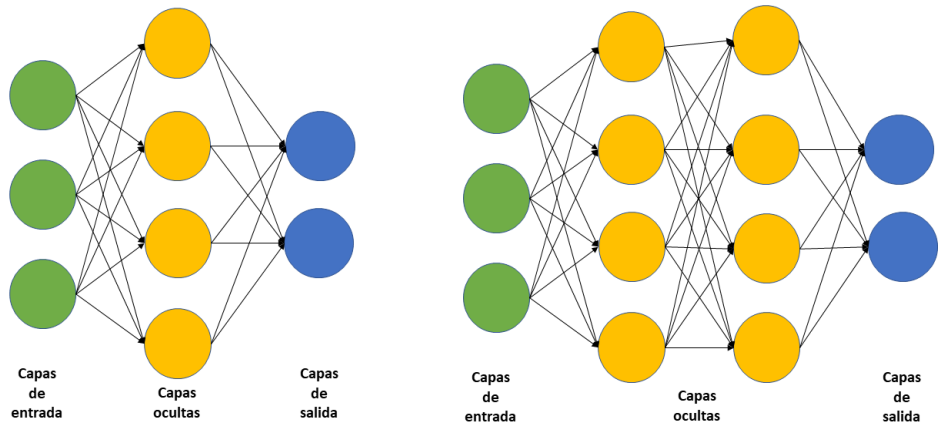


Figura 2.2: Estructura del microscopio electrónico de barrido.

2.3 Redes neuronales artificiales

Una red neuronal consta de varias capas y nodos que reciben entradas de otras capas y producen una salida hasta obtener un resultado final. Las redes neuronales pueden tener cualquier número de capas ocultas. Cuantas más capas tenga una red, más compleja será. Normalmente, las redes neuronales tradicionales constan de 2 o 3 capas ocultas, mientras que las redes de aprendizaje profundo pueden tener un mayor número de capas ocultas. Un ejemplo sencillo se muestra en la Fig. 2.3, donde se ilustra un ejemplo de la estructura que tiene este tipo de redes. Dicho lo anterior, las redes neuronales artificiales (ANN, por sus siglas en inglés) son una de las arquitecturas dentro del aprendizaje profundo que más atención han recibido en la investigación académica y en muchos campos, en diferentes aplicaciones. Las redes neuronales se componen de capas de entrada, intermedias y de salida, formadas por neuronas. Estas neuronas funcionan de forma similar a las neuronas del cerebro humano, permitiendo la transferencia de información de una capa a otra en distintos grados, en función de la predicción final. Los valores que determinan el grado de transferencia de información entre capas se denominan pesos.



(a) ANN simple. (b) ANN profunda.

Figura 2.3: Modelos de ANN.

Los pesos son los enlaces que existen entre neuronas dentro de una ANN, encargados de determinar la importancia de los datos de entrada y realizan una tarea esencial en el rendimiento de la red.

Cuando se diseña una ANN, los pesos se inicializan de manera aleatoria. A medida que la red se entrena, estos pesos se ajustan para resolver el problema, con el objetivo de minimizar el error entre las predicciones de la red y los datos reales. Este proceso se conoce como retropropagación y es el mecanismo que permite que la red aprenda de los datos.

Los valores que se le dan a los pesos en una ANN definen la capacidad de la red para aprender patrones o tendencias en los datos. Si los pesos son demasiado pequeños, la red no puede captar la complejidad de los datos; en caso contrario, si estos son demasiado grandes, la red puede sobreajustarse a los datos de entrenamiento y funcionar mal con datos nuevos. Existen formas de evitar el sobreajuste con técnicas de regularización, que se pueden lograr agregando nuevas capas, como las dropout.

La regularización es una técnica que penaliza los pesos grandes y anima a la red a utilizar pesos más pequeños. Esto puede mejorar la capacidad de la red para generalizar con nuevos datos.

Las ANN destacan en diversas aplicaciones, como el reconocimiento de patrones. Utilizan redes de perceptrón para realizar cálculos con el fin de reconocer características, lo que constituye una forma de aprendizaje supervisado. Además, hacen uso de redes de retropropagación para ajustar los pesos o valores en las conexiones entre neuronas, reduciendo así los errores de predicción en un modelo. También se suelen utilizar máquinas de aprendizaje, que se basan en ecuaciones

matemáticas sencillas. Las redes de perceptrón son especialmente eficaces cuando se trabaja con conjuntos de datos linealmente separables, con un número finito de soluciones, ya que se basan en los coeficientes del hiperplano para separar las clases. Sin embargo, pueden no ser adecuadas para todos los tipos de problemas en este ámbito. En general, estos métodos han sido ampliamente investigados y han demostrado su eficacia en el reconocimiento de patrones [20].

En el ámbito de las redes de retropropagación, un concepto destacado derivado del perceptrón es el perceptrón multicapa. Entre estos, sobresalen los algoritmos basados en la regla delta, que redefinen el funcionamiento de las redes neuronales. A falta de un lenguaje específico, han surgido diversas variaciones algorítmicas. Uno de estos métodos es el de la regla de los mínimos cuadrados. Su ecuación representa el error empírico denotado por la ecuación 2.1 y se aplica a redes perceptrón multicapa.

$$R_{\text{emp}}(w) = \sum_{i=1}^{n} \left[F(x^{(i)}, w) - y^{(i)}\right]^2 \tag{2.1}$$

Dicha formula se utiliza para calcular la función de error empírico (también conocida como función de costo o función de pérdida). En esta expresión, w representa los parámetros o pesos del modelo, $x^{(i)}$ son las características (entradas) del i-ésimo ejemplo, y $y^{(i)}$ es el valor real o objetivo correspondiente a ese ejemplo. $F(x^{(i)}, w)$ es la predicción del modelo para el i-ésimo ejemplo, basada en las entradas $x^{(i)}$ y los pesos w. Finalmente, n es el número total de ejemplos en el conjunto de entrenamiento. La fórmula calcula la suma de los errores cuadrados entre las predicciones y los valores reales para todos los ejemplos, lo que permite evaluar el desempeño del modelo. Estas redes intentan resolver problemas complejos mediante algoritmos adaptativos que les permiten aprender de los errores y mejorar continuamente. Dentro de las redes neuronales, existen variables que ayudan a optimizar su rendimiento. Algunas de estas variables, como los hiperparámetros, son fundamentales para mejorar el desempeño de la red, especialmente en el contexto de la clasificación.

2.3.1 Activación de una neurona

En una ANN la activación es el proceso mediante el cual una neurona calcula su salida a partir de las entradas ponderadas y un sesgo, utilizando una función de activación que introduce no linealidad y permite que la red aprenda patrones complejos. Este concepto está inspirado en el funcionamiento de las neuronas biológicas, que son las unidades fundamentales de las redes neuronales en los sistemas biológicos. A nivel biológico, una neurona recibe un impulso eléctrico proveniente de otras neuronas, transportando información con diferentes

intensidades. Cuando la intensidad del impulso es alta, la neurona se estimula, lo que la lleva a un estado de excitación y genera una respuesta, pensamiento o idea. Si la intensidad es baja, la neurona permanece inactiva, lo que es similar a cuando una persona está pensando sin emitir una respuesta inmediata. Siguiendo este principio teórico, se diseñó un modelo de red neuronal artificial, como se muestra en la Fig. 2.4. Cada neurona en este modelo realiza un promedio ponderado de

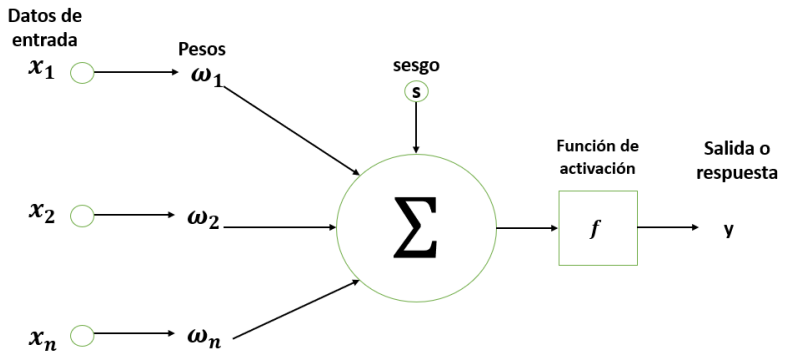


Figura 2.4: Activación en una ANN.

los datos de entrada, sumando un sesgo y aplicando una función de activación para transmitir la información generada por la combinación lineal de los pesos y las entradas. En tareas más complejas, las redes neuronales se organizan en capas, lo que facilita las operaciones lineales de sumas ponderadas. Para ejecutar funciones no lineales y resolver problemas más complicados, se emplean funciones de activación, lo que permite generar resultados no lineales y, por lo tanto, ejecutar tareas más avanzadas (Fig. 2.5). La ecuación representa la combinación lineal de

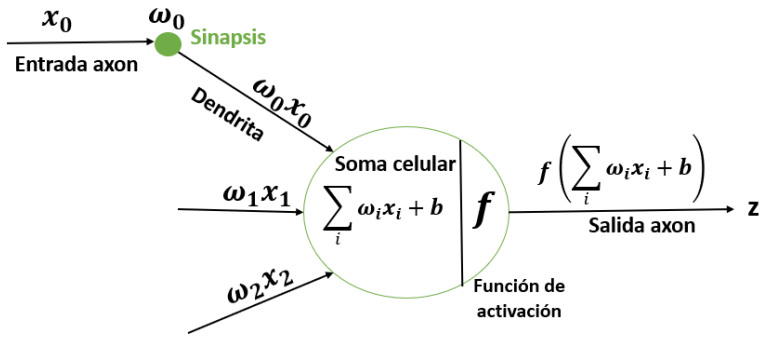


Figura 2.5: Operación lineal de suma ponderada.

las entradas x_i, ponderadas por los pesos w_i y ajustadas por el sesgo b. El valor z se pasa luego a través de una función de activación para obtener la salida final de la neurona.

2.4 Aprendizaje Profundo

El Aprendizaje Profundo (DL, por sus siglas en inglés) es una rama que utiliza técnicas avanzadas basadas en el Aprendizaje Automático (ML, por sus siglas en inglés). Los sistemas que emplean DL muestran un mejor rendimiento, y su único requerimiento esencial es el acceso a grandes cantidades de datos (como imágenes), lo que les permite llevar a cabo tareas complejas a partir de procesos más simples. A su vez, tanto DL como ML forman parte de una disciplina más amplia: la Inteligencia Artificial (IA) como se muestra en Fig. 2.6. Estas áreas ofrecen un gran campo de conocimiento, siendo, en la actualidad, prometedoras y con un crecimiento constante, gracias a los avances tecnológicos y al manejo de grandes cantidades de datos. Los modelos de DNN son una de las principales herramientas que facilitan el trabajo con estos datos.

La IA tiene una gran variedad de aplicaciones y continúa siendo un campo muy dinámico (en constante cambio). Un ejemplo de ello es su impacto en la investigación, al permitir la creación de sistemas inteligentes y la automatización de procesos. En la segunda mitad del siglo XX, específicamente en los años 50, la IA comenzó a ser considerada una disciplina relevante. Gracias a sus contribuciones, se establecieron las bases del DL, tales como el desarrollo de asistentes virtuales [21].

En sus primeros días, la IA se enfocó en resolver problemas que, aunque complejos o tediosos para los humanos, eran relativamente sencillos para las computadoras. Esto se lograba mediante enfoques matemáticos y fórmulas que facilitaban la resolución de dichos problemas. El desafío al que se enfrenta este trabajo de tesis en DL es la resolución del problema de clasificación de imágenes, un área que ha sido abordada por otras líneas de investigación, como la visión por computadora. Además, como destaca el autor de [22], se pueden describir teorías que permiten lograr que un equipo de cómputo alcance una solución con claridad. Sin embargo, es fundamental que el usuario defina de manera precisa los pasos a seguir, partiendo de conceptos y técnicas que, aunque a veces complejas, se construyen a partir de otras más sencillas. Esto permite, a su vez, construir las denominadas "capas" en una red neuronal, de las cuales se hablará con más detalle en secciones posteriores, siendo estas uno de los fundamentos del DL.

La principal novedad en el DL es que contiene capas ocultas o intermedias, tomadas desde el enfoque de las ANN, cuyo principio inicial se creó a partir de las neuronas a niveles biológicos [23]. Se parte del entrenamiento de un conjunto de datos dentro de una red, con la finalidad de que esta aprenda a reconocer características y tenga la capacidad de resolver problemas. Para que el DL realice un buen entrenamiento y se logre el producto deseado, es necesario contar con un conjunto de datos que posea ciertas características ideales, como la variación del

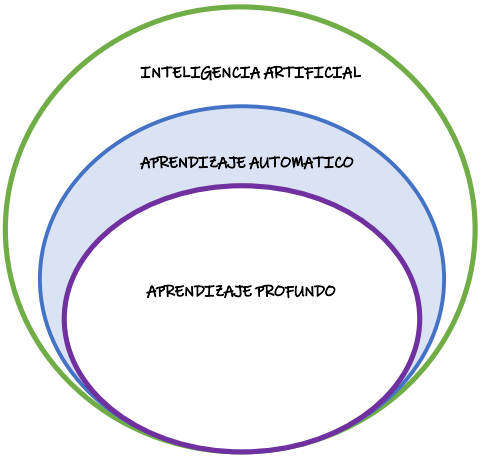


Figura 2.6: Subconjunto de la disciplina de inteligencia artificial.

punto de vista, la iluminación y una calidad gráfica variada. Esto es fundamental si nuestra red aprende a clasificar imágenes, ya que permite que la red tenga la posibilidad de realizar clasificaciones más precisas. Por ejemplo, consideremos la tarea de separar el fondo de una imagen que contiene un coche. Partimos de una imagen en color formada por tres canales de color, cada uno de ellos compuesto por píxeles. Para ello, clasificaremos la imagen en dos categorías: una para el fondo y otra para el coche, incluyendo sus características e información distintivas. Estas características deben obtenerse mediante métodos de preprocesamiento bien conocidos en el campo de la extracción de características.

Por otra parte, si el conjunto de datos no presenta una buena calidad en las imágenes o no tiene mucha variedad de clases, puede dar un resultado con muy baja latencia de acierto. Puede que nuestra red esté en forma óptima, pero si sus datos son de mala calidad, no cumplirá con su propósito, presentando errores y confusiones con el resto de los datos. En la actualidad, los avances tecnológicos han agilizado los procesos y las tareas diarias, estando al alcance de los usuarios, quienes disfrutan de manera indirecta de los beneficios que estos avances, como pueden ser: el reconocimiento del movimiento en un paciente con hemiparesia [24], otros casos en el reconocimiento de autos [25], y sistemas de seguridad [26], entre muchos otros. El DL se subdivide en varias categorías, cada una desarrollada para resolver de forma específica diferentes tareas, siendo las redes neuronales convolucionales (CNN, por sus siglas en inglés) las que presentan mayor enfoque en la clasificación de imágenes.

2.4.1 Redes neuronales profundas

Una DNN es una técnica de aprendizaje automático que permite a un ordenador realizar tareas complejas que serían difíciles de lograr utilizando métodos de programación convencionales. Están diseñadas para funcionar no solo siguiendo reglas preestablecidas, sino que también tienen la capacidad de predecir soluciones y extraer conclusiones basadas en el número de iteraciones que realizan. Permiten capturar las características de las imágenes de manera efectiva, comenzando con los filtros de convolución, que son responsables de extraer los datos. Esto se logra gracias a la incorporación de capas ocultas; cuanto mayor sea el número de capas, mayor será el grado de complejidad [27].

2.4.2 Redes neuronales convolucionales

Una CNN de clasificación toma una imagen de entrada y genera una distribución de porcentajes de clase, a partir de la cual se puede encontrar la clase con mayor probabilidad. Está compuesta por varias capas. Un ejemplo de ello se muestra en la Fig. 2.7.

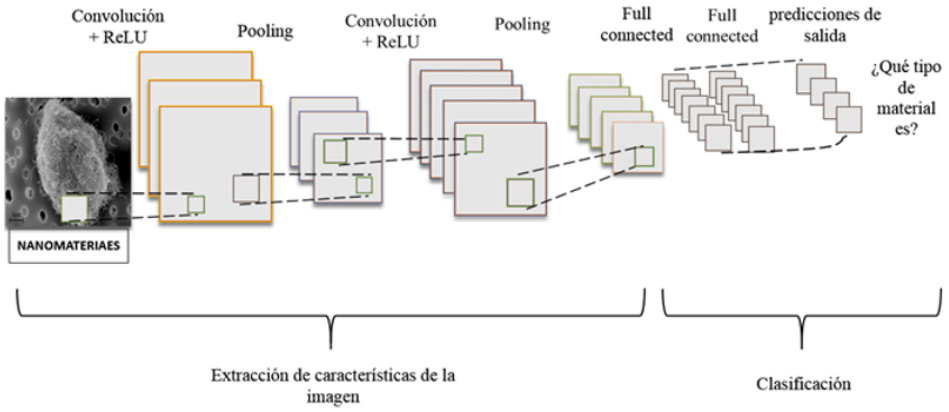


Figura 2.7: Composición de las capas en una CNN.

Las capas de extracción de características de las imágenes de entrada son las capas ocultas, y están encargadas de la tarea de reducir la dimensionalidad de entrada mediante cálculos y, eventualmente, producir un resultado de porcentajes de clase. Las CNN están compuestas por múltiples capas que tienen un orden, en las que destacan tres tipos principales de capas: convolución, agrupación y totalmente conectadas. Cada una de estas se compone de nodos, es decir, un punto de interconexión o de aprendizaje en el que se generan operaciones según sea el caso a resolver. Las capas de convolución son las encargadas de realizar la extracción de características, es decir, encontrar características especiales en

una imagen de entrada. Aquí es donde se generan los filtros, o también llamados núcleos convolucionales. El desplazamiento en una imagen es una acción llamada paso o stride, que se mueve píxel a píxel, produciendo una imagen de salida de aproximadamente el mismo tamaño que la imagen de entrada. Al utilizar múltiples filtros, se generan imágenes de salida mucho más pequeñas.

Etapas de convolución

En esta sección, explicaremos las etapas de convolución necesarias para realizar la tarea de clasificación utilizando una CNN. Una CNN es una arquitectura multicapa que consta de capas con convoluciones, funciones de activación no lineales y unidades lineales, además de existir también parámetros a considerar, como los llamados hiperpárametros. Cada capa tiene un número determinado de convoluciones.

Funciones de activación

A continuación se presenta una descripción de las funciones de activación más comunes, incluida la unidad lineal rectificada utilizada en este trabajo [28].

- **La unidad lineal rectificada** (ReLU, por sus siglas en inglés) es una función que convierte los valores de entrada de un nodo en valores de salida. En las ANN, se simula el proceso biológico que ocurre en el soma (cuerpo); al recibir señales de las sinapsis, se genera un potencial de acción cuando se supera un umbral determinado. La ReLU se destaca por su capacidad de reemplazar todos los valores negativos de los píxeles en el mapa de características por cero. Esta función tiene las siguientes características:

  Gradiente saturado: Por regla general, todos los valores son mayores que cero. Esto evita los problemas de retropropagación y permite actualizar rápidamente los parámetros en la primera capa.

  Baja complejidad computacional: Durante el proceso de retropropagación, un gradiente negativo elevado puede dificultar el aprendizaje. Esto ocurre porque la derivada de la neurona de entrada es cero cuando su valor es menor que cero, lo que resulta en un gradiente nulo. Para solucionar este problema, se pueden ajustar los pesos o emplear Leaky ReLU (LReLU, por sus siglas en inglés), una variante del ReLU que permite una pequeña salida lineal multiplicada por un valor cercano a 0.001, evitando así que el gradiente se vuelva cero.

- **La función tangente hiperbólica** (tanh) es una variante sigmoidea que proporciona salidas reales dentro de un rango definido. Sin embargo, suele

sufrir saturación de gradiente.

$$tanh(x) = 2sigmoid(2x) - 1 \tag{2.2}$$

La ecuación 2.2 muestra cómo la tangente hiperbólica (tanh) puede expresarse en términos de la función sigmoide. La sigmoide, que mapea valores entre 0 y 1, se escala y centra al multiplicarla por 2 y restarle 1, lo que transforma su rango a entre -1 y 1, igual que la función tangente hiperbólica. Esta relación resalta cómo ambas funciones están estrechamente relacionadas.

- **La función sigmoide:** es una función de activación clásica que depende del signo del parámetro en la ecuación 2.3. Su forma se ajusta hacia la izquierda o hacia la derecha, lo que la hace adecuada para modelar conceptos que implican incrementos o decrementos. Esta posee dos problemas que provocan una convergencia lenta de los parámetros, afectando la eficiencia en el entrenamiento de nuestro modelo:

  Saturación del gradiente: cuando el valor dado por la función se aproxima a valores entre 0 y 1, el gradiente de dicha función tiende a 0, lo que afecta el ajuste de los pesos utilizados por las redes.

  Pesos positivos de forma continua: el valor medio de la función de salida no es 0, lo que origina que los pesos tiendan a ser positivos.

  $$f(a, x, c) = 1/1 + e^{-}a(x - c) \tag{2.3}$$

  La ecuación 2.3 describe una función sigmoide que es ajustable mediante los parámetros a (pendiente) y c (desplazamiento). Esta función se usa ampliamente en modelos de aprendizaje automático, especialmente para clasificación binaria, al mapear cualquier valor real de x al rango entre 0 y 1.

- **La función de activación softmax** convierte las salidas de una capa de neuronas en una distribución de probabilidad, y se utiliza comúnmente en tareas de clasificación multiclase. Su propósito es transformar los valores de salida de las neuronas en probabilidades, de manera que la suma de todas ellas sea igual a 1. Esto es particularmente útil cuando se necesita predecir una clase dentro de un conjunto de clases posibles. Dada una entrada z_i, la salida de la función softmax se calcula con la ecuación 2.4.

  $$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{n} e^{z_j}} \tag{2.4}$$

  Donde z_i es la salida de la i-ésima neurona, y $\sum_{j=1}^{n} e^{z_j}$ es la suma de las exponenciales de todas las salidas. Las salidas son valores entre 0 y 1, y su suma es igual a 1, lo que las convierte en probabilidades.

Capas

La aplicación de la convolución en imágenes implica el uso de matrices 2D, llamadas filtros o kernels, que recorren la imagen a clasificar, como se muestra en la Fig. 2.8. Durante la operación de convolución, los filtros se aplican a una matriz de píxeles que conforma la imagen, solapándose entre cada paso. La amplitud de este desplazamiento se conoce como stride (del español, paso). En algunos casos, es conveniente rellenar la matriz de entrada con ceros en los bordes, lo que permite aplicar el filtro a los elementos cercanos al borde de la imagen. Este relleno con ceros ayuda a controlar el tamaño de los mapas de características. Se le conoce como convolución amplia cuando se utiliza relleno, y convolución estrecha cuando no se emplea relleno [29].

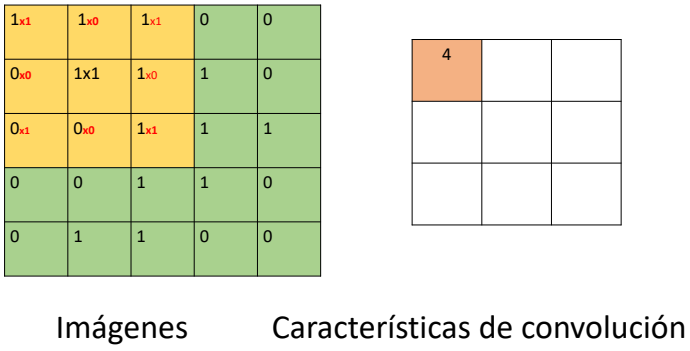


Figura 2.8: Aplicación de la convolución. Ejemplo de convolución en una imagen de 5x5 utilizando un filtro de 3x3 píxeles y un stride de 1, la salida resultante es una imagen de 3x3 píxeles.

Operación de convolución

También conocida como extracción de características, la matriz de la izquierda representa una imagen en blanco y negro, donde cada entrada corresponde a un píxel, asignando valores de 0 para negro y 1 para blanco (aunque comúnmente se utiliza una escala de 0 a 255 para imágenes en escala de grises).

La ventana que se desliza sobre la imagen se denomina kernel, filtro o detector de características. La región de la matriz de la imagen de entrada, de tamaño igual

al del filtro, se conoce como campo receptivo, como se muestra en la Fig. 2.9.

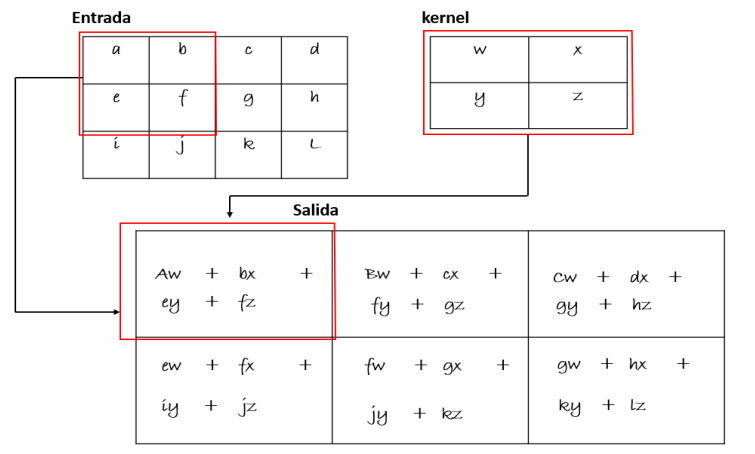


Figura 2.9: Visualización de convolución, desplazamiento.

Los valores del filtro se multiplican por los correspondientes en el campo receptivo, elemento por elemento, y luego se suman en cada paso. El filtro se desliza sobre el siguiente campo receptivo de la matriz y repite el proceso hasta cubrir toda la imagen. El resultado de esta operación es un valor que representa el volumen de la salida, el cual se pasa a la siguiente capa. Las capas convolucionales buscan activar la función ReLU, que convierte los valores negativos en cero [30].

Capa de agrupación

Cuando trabajamos con CNN, utilizamos una técnica llamada pooling para reducir el volumen espacial de una imagen. Desde una perspectiva física, puede considerarse como una ventana que agrupa píxeles y realiza una operación específica en la imagen para reducir su dimensionalidad. En cada stride, la posición de la ventana se actualiza en función de sus desplazamientos (de los strides); este se da entre ventanas.

Una operación común de agrupación es la operación de máximo (max), que divide la entrada en ventanas y emite el valor máximo de cada ventana. La agrupación máxima suele realizarse entre dos capas de convolución, ya que aplicarla a una capa totalmente conectada sería demasiado costoso [31].

El max-pooling permite reducir la entrada sin perder información espacial. Un ejemplo de este desplazamiento se puede ver en la Fig. 2.10.

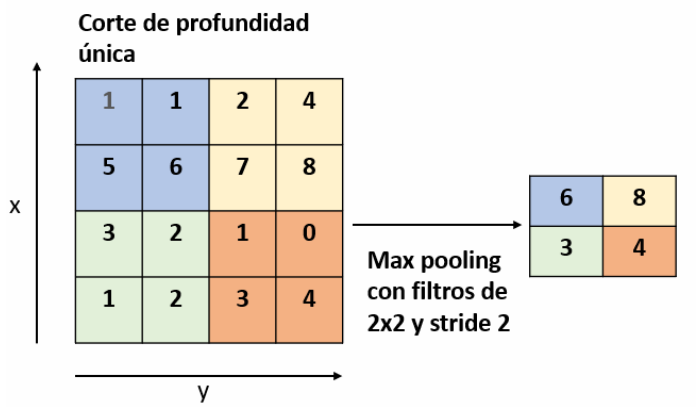


Figura 2.10: Al aplicar un filtro 2x2 con un stride de 2 píxeles, la imagen se reduce de 4x4 a 2x2.

Capas totalmente conectadas

Al final de una CNN, se suelen situar las capas totalmente conectadas (FC, por sus siglas en inglés). Dependiendo de la arquitectura concreta que se utilice, la ubicación de estas FC puede variar. Actualmente, existen muchas redes preentrenadas disponibles para experimentar cuando se trabaja con bases de datos de imágenes. Una de estas redes es AlexNet, que consta de 5 capas convolucionales y 3 capas FC. En la mayoría de los casos, estas capas FC se colocan después de una capa convolucional [32].

Las FC son muy eficientes, según la literatura [33], y dependen de la profundidad de la arquitectura de una CNN. Estas se utilizan para aprender combinaciones no lineales de características obtenidas a lo largo de una etapa de entrenamiento.

Capa de salida

Esta capa tendrá tantos nodos como clases queramos clasificar. El resultado de la probabilidad de pertenencia a dicha clase será el que aparezca en este nodo.

Clasificación de imágenes

En una CNN, la última capa se denomina capa de clasificación. Su propósito es clasificar la entrada en clases mutuamente excluyentes (cada elemento solo puede pertenecer a una de las clases), basándose en las probabilidades de la función de activación, para luego calcular la pérdida. Cuando se trata de la clasificación de imágenes, los pesos asignados a cada nodo en las capas de la red neuronal son

inicialmente aleatorios, pero desempeñan un papel crucial en la determinación de nuestras predicciones. Se ejecuta un complejo procedimiento que incluye recorrer la red en sentido inverso (es decir, un proceso de retropropagación).

Durante este proceso, se recalculan los pesos y filtros de la red para ajustarlos y minimizar los errores. Para medir el error, utilizamos una función de costo (una función de pérdida), que se calcula como la diferencia entre la salida predicha y la salida real. El objetivo es minimizar la función de costo seleccionando la más adecuada para la tarea específica a tratar. Para ello, hay que definir la arquitectura de la red neuronal, incluyendo el número de capas y la función de costo. La función de costo mide el error cometido por el modelo y es lo que pretendemos minimizar (en el caso de la clasificación, entropía cruzada). Una vez definido el modelo, podemos empezar a entrenarlo utilizando las imágenes del conjunto de entrenamiento. Cada imagen del conjunto de entrenamiento se procesa aplicando filtros (lo que se conoce como convolución) y otras técnicas para producir nuevas imágenes que pasan por capas sucesivas. Las imágenes generadas en la última capa se envían a un clasificador totalmente conectado, que determina la clase a la que pertenece la imagen.

Este proceso de presentar el conjunto de entrenamiento y recalcular la red se repite hasta que se alcanza un error aceptable o hasta que el error ya no puede reducirse. Para que una CNN aprenda adecuadamente, es necesario entrenarla sucesivamente para que calcule los filtros y clasificadores adecuados, y así ajustar nuestro modelo en función de los mejores parámetros dados. Entrenar una CNN es un proceso que requiere experiencia y experimentación. En la Fig. 2.11 se presenta un diagrama del proceso de clasificación de imágenes.

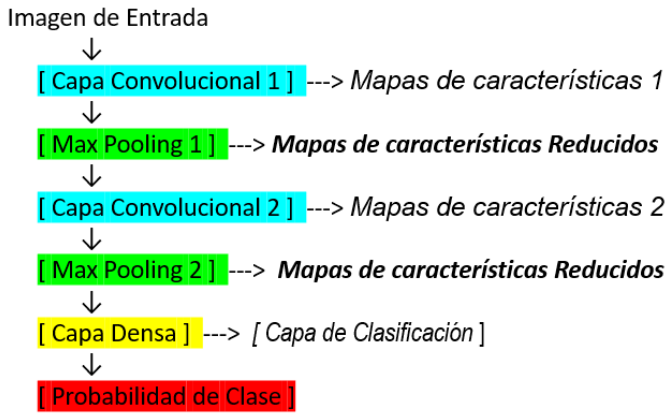


Figura 2.11: Diagrama simplificado del proceso de clasificación de imágenes en una CNN

Este diagrama ilustra de manera simplificada el flujo de información a través

de una red neuronal convolucional para la clasificación de imágenes. La imagen de entrada pasa a través de varias etapas de procesamiento:

- Capa Convolucional 1: Aplica filtros para extraer características básicas de la imagen, como bordes y texturas.
- Max Pooling 1: Reduce la dimensionalidad de la imagen, manteniendo solo las características más importantes.
- Capa Convolucional 2: Detecta patrones más complejos, como formas y objetos, a partir de las características extraídas en la capa anterior.
- Max Pooling 2: Reduce aún más el tamaño de las representaciones, destacando las características esenciales.
- Capa Densa: Las características extraídas por las capas convolucionales se combinan en una capa densa que realiza la clasificación final.
- Clasificación: Finalmente, la red asigna una probabilidad a cada clase posible, determinando la categoría de la imagen (por ejemplo, gato, perro, coche, etc.).

Este flujo iterativo y jerárquico de procesamiento, permite que la red neuronal extraiga representaciones cada vez más complejas, simplificando el proceso de clasificación de imágenes.

En resumen, el producto de la convolución y la matriz resultante permiten que las CNN realicen la clasificación de imágenes, destacando las características clave mientras reducen la complejidad de los datos originales. Este proceso iterativo es fundamental para que una CNN aprenda y mejore continuamente su capacidad de clasificación, lo que la convierte en una herramienta poderosa en el campo del DL.

2.4.3 Arquitecturas preexistentes

VGG16

Se trata de un modelo de CNN presentado en el ImageNet Challenge, un evento anual que exhibe y pone a prueba modelos de visión por ordenador. En 2014, el Departamento de Ciencias de la Ingeniería de la Universidad de Oxford presentó esta arquitectura, que obtuvo el primer y segundo puesto en la categoría de detección y clasificación de objetos. Esta arquitectura se distinguió por su diseño simple y profundo, utilizando bloques de convolución con filtros de tamaño 3x3, lo que resultó en una mejora significativa. La mayor profundidad se logró gracias a la implementación de dos arquitecturas, una con 16 capas y otra con 19 capas,

lo que resultó en aproximadamente 138 millones de parámetros entrenables [34]. VGG16 se compone de 16 capas, 13 de las cuales son convolucionales, y las tres restantes son densas. El término *VGG16* se deriva del número de capas de la red. Fue diseñada para clasificar hasta 1000 categorías; sin embargo, hemos modificado la capa final con el fin de clasificar solo tres categorías, tal como se muestra en la Fig. 2.12. Esta red es un recurso ideal para tareas de clasificación que requieren menos recursos computacionales y memoria.

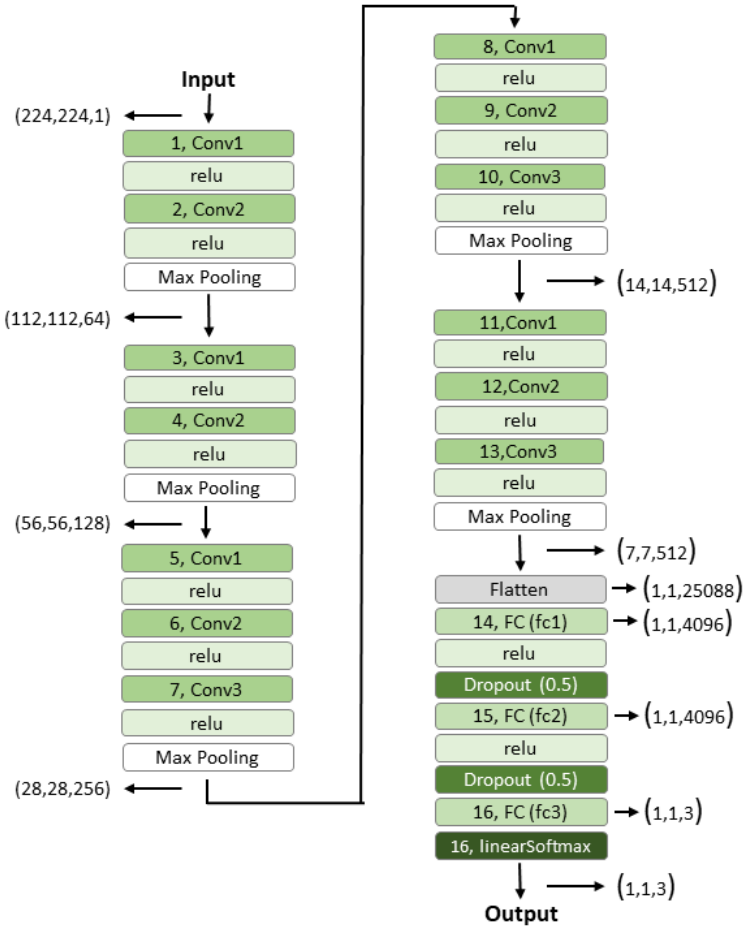


Figura 2.12: Arquitectura de red VGG16.

AlexNet

Otra de las arquitecturas utilizadas, y ganadora del concurso ImageNet en 2012, introdujo el uso de CNN profundas y la función ReLU para mejorar el rendimiento.

Esta arquitectura se muestra en la Fig. 2.13. Este modelo funciona con 60 millones de parámetros y 65,000 neuronas. Fue creado por Alex Krizhevsky en ese mismo año [35]. AlexNet ha incorporado ocho capas: cinco capas convolucionales, dos capas ocultas completamente conectadas y una capa de salida completamente conectada. Se usa la función de activación ReLU en lugar de opciones tradicionales como logistic, tanh y arctan, debido a que ayuda a evitar el problema del desvanecimiento del gradiente en los modelos de DL. También se utilizan capas de abandono (del inglés, dropout), un concepto que evita eficazmente el sobreajuste, modificando la estructura de la red neuronal. Para una capa determinada de neuronas, algunas neuronas se eliminan aleatoriamente con una probabilidad determinada, manteniendo inalteradas las neuronas de la capa de entrada y de la capa de salida, y actualizando posteriormente los parámetros de acuerdo con el método de aprendizaje de la red neuronal. En una posterior iteración, se eliminan de manera aleatoria algunas neuronas hasta el inicio del entrenamiento. AlexNet, en general, es conocida por sus grandes capacidades en la clasificación de imágenes, según la literatura [36].

ResNet50

Una DNN, creada en el año 2015 con el fin de solventar dificultades en cuanto a la clasificación de imágenes, se compone de 50 capas, 16 bloques residuales, y cada bloque contiene múltiples capas convolucionales con conexiones de salto. La arquitectura también comprende capas de agrupamiento, capas totalmente conectadas y una capa de salida softmax para la clasificación, como se muestra en el diagrama de arquitectura de la Fig. 2.14.

Algunas variantes de esta arquitectura son ResNet101 y ResNet152. El modelo ResNet50 se utiliza comúnmente como red preentrenada para la clasificación de grandes volúmenes de imágenes. Los resultados de ResNet han mostrado ser muy prometedores en aplicaciones de imágenes médicas, y su potencial para ser implementada en otros tipos de imágenes es considerable [37].

ResNet es un acrónimo de red residual; es una arquitectura que ha ganado popularidad en el campo del reconocimiento y clasificación de imágenes. Las redes neuronales convolucionales profundas (DCNNs, por sus siglas en inglés) se han utilizado ampliamente en diversas aplicaciones, pero la tendencia actual es emplear redes más profundas para abordar tareas más complejas y mejorar la precisión del reconocimiento. Sin embargo, desarrollar redes más profundas presenta desafíos, como la disminución del gradiente y la degradación del rendimiento. El aprendizaje residual fue diseñado para superar estos problemas. En las DNN tradicionales, el gradiente evanescente ocurre cuando los gradientes se vuelven extremadamente pequeños al propagarse a través de múltiples capas. A medida que aumenta el número de capas, el error en el entrenamiento o la prueba también aumenta, lo que

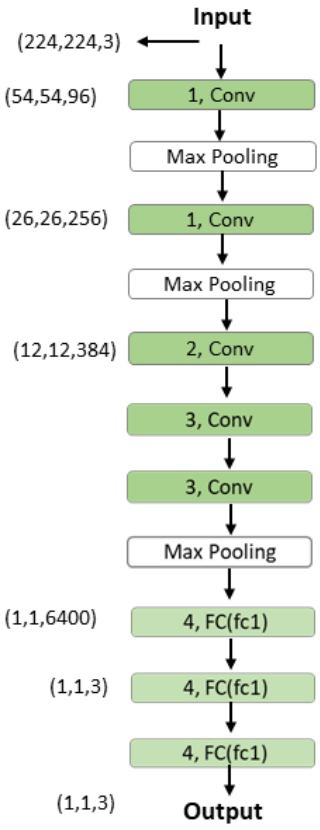


Figura 2.13: Arquitectura de red AlexNet.

ralentiza el aprendizaje o incluso impide el aprendizaje en redes muy profundas.

Conexiones de salto

El modelo ResNet50 aborda el problema del gradiente evanescente mediante la incorporación de conexiones de salto , lo que permite construir redes muy profundas sin que se presenten problemas de desvanecimiento del gradiente. Cuando las redes más profundas intentan converger, surge un problema de degradación: a medida que aumenta la profundidad, la precisión se estanca (lo que podría ser esperado) y luego disminuye rápidamente. Sorprendentemente, esta degradación no se debe a un sobreajuste. Al agregar más capas a una red lo suficientemente profunda, se genera un mayor error de entrenamiento [38].

La disminución de precisión durante el entrenamiento indica que algunos sistemas son más difíciles de optimizar que otros. Para abordar este problema, se implementa el mapa residual o conexión residual, que consiste en eliminar

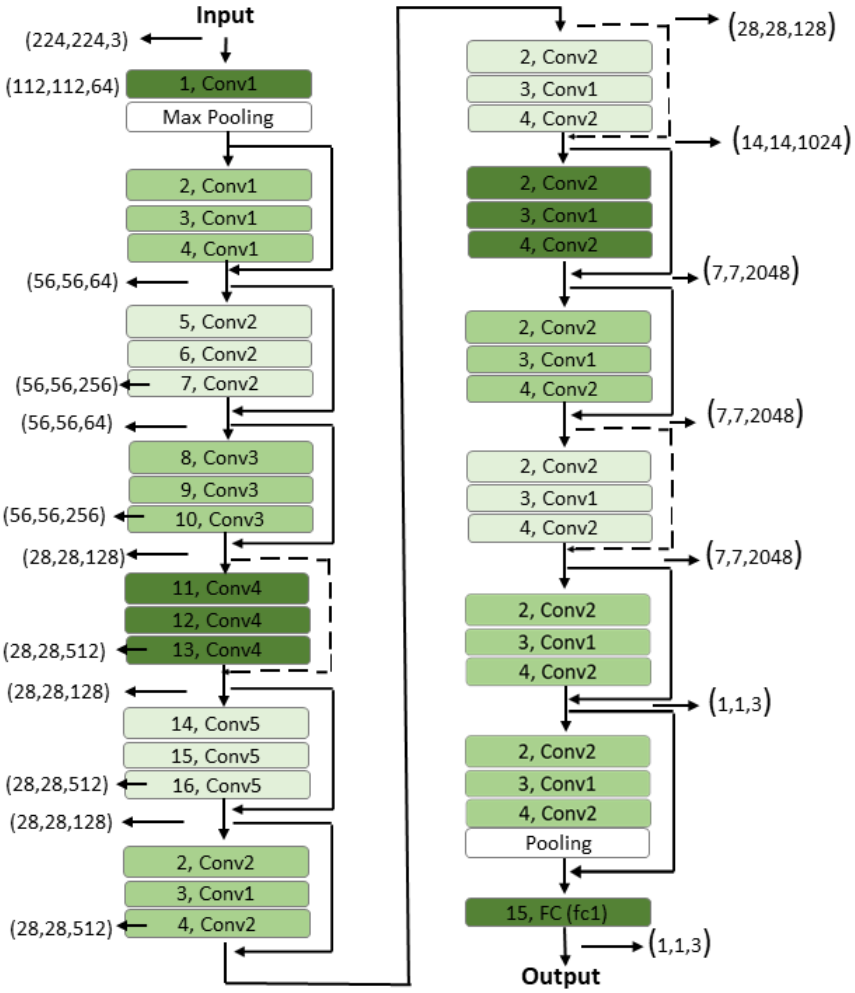


Figura 2.14: Arquitectura de red ResNet50.

los bloques de convolución entre dos unidades de activación ReLU y conectar directamente la salida de la primera ReLU a la segunda. Con esta conexión residual, la red aprende la función residual en lugar de la función completa, lo que facilita la optimización. Aprender las diferencias residuales entre la entrada y la salida es más sencillo que aprender la función completa desde cero, lo que hace que el proceso de aprendizaje sea más eficiente, especialmente en redes profundas. Añadir capas adicionales no afecta negativamente el rendimiento, ya que la regularización y los pesos se omiten en ese momento. Esto asegura que el rendimiento no se degrade al aumentar el número de capas de 20 a 50 o de 50 a 100; basta con agregar un bloque de identidad, como se muestra en la Fig. 2.15. Esto permite que los gradientes se propaguen con mayor facilidad desde la capa

de salida hasta las primeras capas, facilitando la implementación de tareas más complejas y mejorando la precisión del modelo.

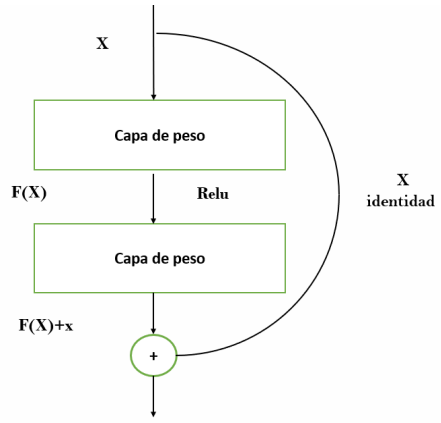


Figura 2.15: Estructura de conexiones de salto. Aquí un ejemplo de cómo se ejecutan las conexiones de salto.

2.4.4 Matriz de confusión

La matriz de confusión es una herramienta que se utiliza en el campo de la IA o el DL para evaluar el rendimiento de un algoritmo de clasificación estadística. véase la Fig. 2.16.

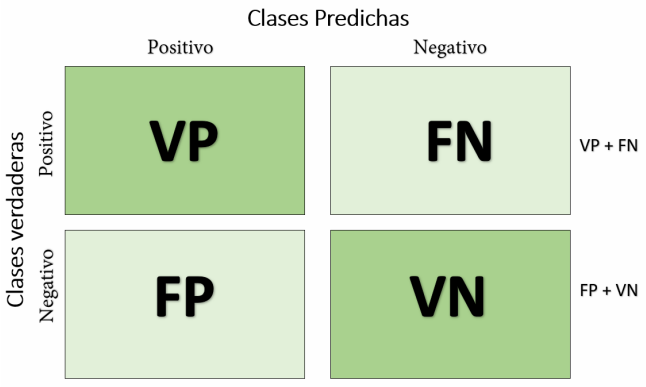


Figura 2.16: Enfoque basado en la matriz de confusión para evaluar el rendimiento del modelo.

En un problema de clasificación binaria se tienen solo dos clases 0 y 1, preferiblemente una clase positiva y otra negativa.

- Verdadero positivo (**VP**): Se refiere al número de predicciones positivas correctas realizadas por el clasificador.
- Verdadero Negativo (**VN**): Se refiere al número de predicciones donde el clasificador predice correctamente la clase negativa como negativa.
- Falso Positivo (**FP**): Se refiere al número de predicciones donde el clasificador predice incorrectamente la clase negativa como positiva.
- Falso Negativo (**FN**): Se refiere al número de predicciones donde el clasificador predice incorrectamente la clase positiva como negativa.

A menudo, las matrices de confusión se utilizan como criterios de evaluación de los modelos DL. Proporcionan medidas de rendimiento muy sencillas pero eficaces. A continuación, se discuten algunas de las medidas de rendimiento más comunes que se pueden utilizar a partir de matrices de confusión. Por mencionar algunos ejemplos de su uso, se puede ver en [39] que se trata de un estudio sobre la detección temprana de enfermedades en plantas.

A diferencia de la clasificación binaria, en un escenario multiclase no hay clases positivas ni negativas. Al inicio, puede resultar difícil identificar los TP, TN, FP y FN, ya que no hay clases positivas o negativas diferenciadas. Sin embargo, el proceso es bastante sencillo. Lo que se debe hacer es identificar los TP, TN, FP y FN para cada clase que se tenga. En capítulos siguientes, se hablará un poco más de esto.

2.4.5 Métricas de evaluación

Aquí se muestran las métricas obtenidas a partir de las matrices de confusión para evaluar el rendimiento de un modelo [40].

Precisión

La precisión es una métrica que indica la proporción de muestras clasificadas correctamente como positivas, es decir, la relación entre los verdaderos positivos y la suma de verdaderos positivos y falsos positivos. En otras palabras, mide la exactitud de las predicciones positivas del modelo.

$$\text{Precisión} = \frac{VP}{VP + FP} \tag{2.5}$$

Exactitud

La exactitud es una métrica que indica la proporción de elementos clasificados correctamente en relación con el total de elementos. Aunque la exactitud es una

medida ampliamente utilizada, tiene limitaciones, especialmente en problemas de clasificación con clases desequilibradas, donde una clase puede dominar el conjunto de datos y el modelo podría clasificar correctamente la mayoría de los casos, pero no identificar adecuadamente las clases minoritarias [41].

$$Exactitud = \frac{VP + VN}{VP + FP + FN + VN} \tag{2.6}$$

Especificidad

La especificidad es una métrica que indica la fracción de las clases negativas que el clasificador identifica correctamente como negativas. Esta medida también es conocida como la tasa de verdaderos negativos (True Negative Rate, TNR, por sus siglas en inglés).

$$Especificidad = \frac{VN}{VP + FP} \tag{2.7}$$

Sensibilidad

La sensibilidad es la proporción de verdaderos positivos con respecto al total de instancias positivas reales en el conjunto de datos. En otras palabras, mide la capacidad del modelo para identificar correctamente los casos positivos.

$$Sensibilidad = \frac{VP}{VP + FN} \tag{2.8}$$

Puntaje-F1

La puntuación F1 es una métrica que representa un equilibrio entre precisión y sensibilidad. Se calcula utilizando la media armónica de la precisión y la sensibilidad, y combina la precisión y la sensibilidad en una única medida. La puntuación oscila entre 0 y 1, siendo 1 la mejor puntuación posible.

$$Puntaje - F1 = \frac{2 * Precision * Sensibilidad}{Precision + Sensibilidad} \tag{2.9}$$

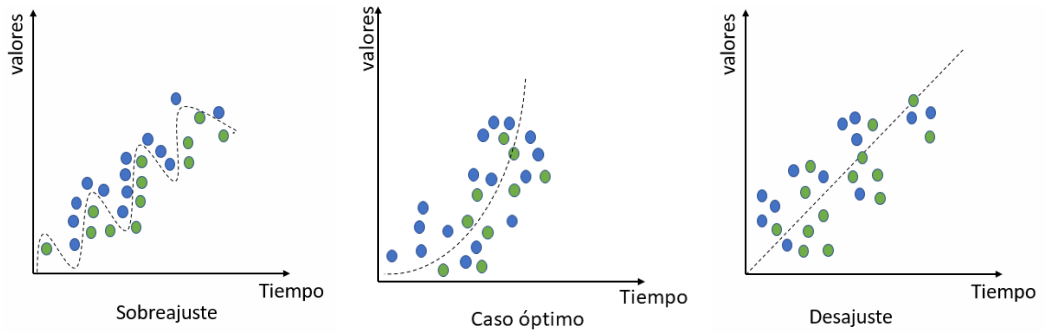
2.4.6 Sobreajuste

El sobreajuste, véase la Fig. 2.17(a). En este caso, el modelo ha aprendido en exceso los detalles de los datos de entrenamiento. Como resultado, el límite de decisión (representado por la línea punteada) se vuelve excesivamente complejo,

siguiendo de cerca las fluctuaciones de los puntos de entrenamiento e incluso separando variaciones que no son representativas. Aunque este modelo presenta un bajo error en los datos de entrenamiento, no logra generalizar bien a nuevos datos, ya que su capacidad para clasificar ejemplos no vistos se ve comprometida por el exceso de ajuste a los datos de entrenamiento.

2.4.7 Desajuste

El desajuste, véase la Fig. 2.17(c). En este escenario, el modelo es demasiado simple para aprender las relaciones complejas entre las características de los datos. El límite de decisión se muestra demasiado rígido o simplificado, lo que impide una separación adecuada de las clases. Como resultado, el modelo no logra capturar las características clave de los datos de entrenamiento, lo que lleva a un alto error tanto en los datos de entrenamiento como en los de prueba. Este tipo de modelo no ha aprendido lo suficiente para generalizar de manera efectiva, lo que da lugar a un rendimiento deficiente.



(a) (b) (c)

Figura 2.17: Casos donde se presenta sobreajuste y desajuste. En esta figura, los puntos verdes y azules representan clases. La línea punteada indica el límite de decisión, que es el criterio que el modelo utiliza para separar las clases. En el caso (a) sobreajuste, el límite de decisión es innecesariamente complejo; en el (b) caso óptimo, el límite es adecuado y balanceado; y en (c) desajuste, el límite de decisión es demasiado simple para separar correctamente las clases. La clave está en encontrar un equilibrio donde el modelo pueda aprender los patrones generales sin ajustarse en exceso o quedarse demasiado general.

2.4.8 Aumentación de datos

Consiste en desarrollar imágenes sintéticas para mejorar la robustez del modelo. Esta técnica se fundamenta en el aumento artificial de imágenes, lo que posibilita la mejora de las imágenes existentes con modificaciones recientes de las originales [42]. Un ejemplo, véase la Fig. 2.18.

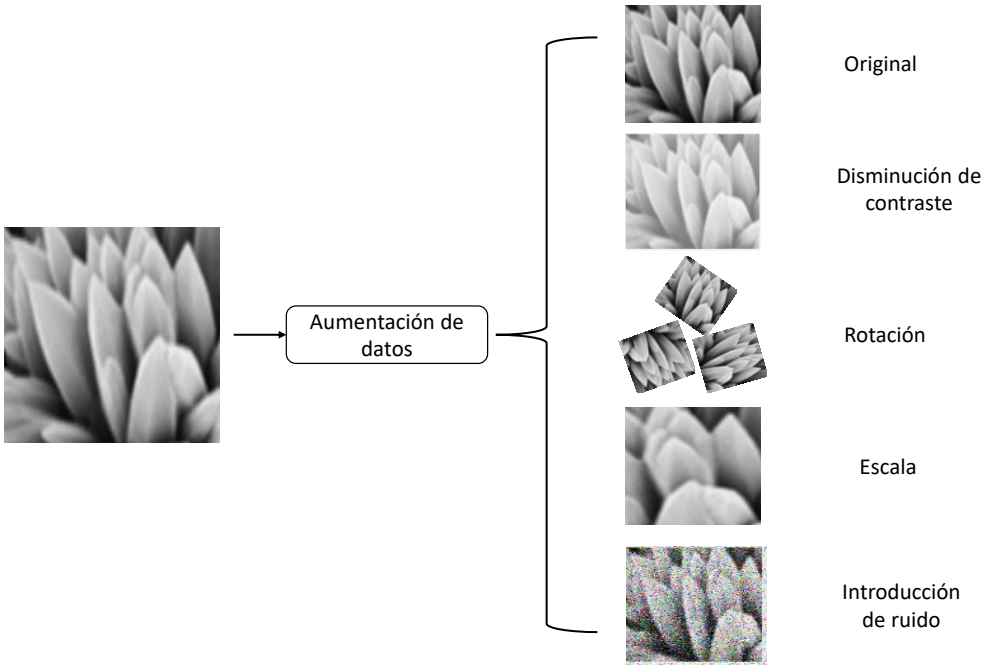


Figura 2.18: Ejemplo de Aumentación de datos. Es una técnica utilizada para expandir el conjunto de datos de entrenamiento mediante la creación de nuevas muestras a partir de las originales.

2.4.9 Ajuste fino

El ajuste fino es una técnica de entrenamiento basada en tomar los parámetros de una red existente con el fin de mejorar su capacidad, adaptando la estructura del modelo existente y entrenándola. Esto consiste en ajustar las últimas capas del modelo para que se adapten a una serie de datos específicos. Se realiza un ajuste ligero para evitar definir una nueva estructura de la red y evitar partir desde cero [43].

2.4.10 Aprendizaje por transferencia

El Aprendizaje por transferencia (TL, por sus siglas en inglés) se suele utilizar cuando se tiene un conjunto de datos limitado. Para hacer uso de esta técnica, se toma un modelo que fue entrenado en una tarea y se reutiliza en otra, congelando los parámetros del modelo ya existente. Se carga el modelo entrenado y se congelan las capas previamente adquiridas, con el fin de prevenir la pérdida de información. Luego, se añaden nuevas capas de aprendizaje sobre las congeladas y se entrenan [44].

2.4.11 Grid search

Muchos modelos de DL tienen varios parámetros que pueden ajustarse para cambiar su funcionamiento. Para evitar el sobreajuste, podemos optimizar estos parámetros mediante un proceso denominado Grid Search (del español, búsqueda en cuadrilla). Esto implica probar diferentes combinaciones de parámetros para encontrar la mejor, que proporcione los resultados más precisos. El método de búsqueda en cuadrilla es sencillo: especificamos una lista de valores para distintos parámetros y, en un equipo de cómputo, se evalúa el rendimiento del modelo con cada combinación de esos parámetros. Esto nos ayuda a encontrar el conjunto óptimo que nos da el mejor rendimiento.

Esta técnica es un procedimiento exhaustivo que cubre todo el espacio de posibles combinaciones de parámetros, pero de una forma espaciada que se asemeja a una cuadrícula. Esto se puede ver en la Fig. 2.19. Aunque la búsqueda en

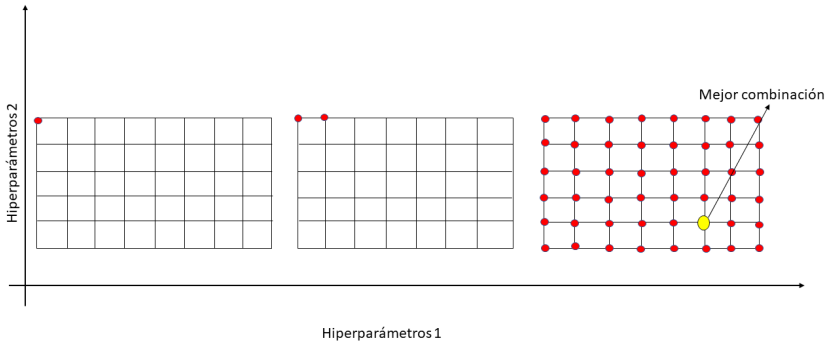


Figura 2.19: Enfoque búsqueda de cuadrilla.

cuadrilla es sencilla de implementar, puede resultar muy costosa debido al gran número de hiperparámetros y a los distintos niveles de cada uno de ellos. Como resultado, el coste computacional aumenta exponencialmente [45].

2.4.12 Validación cruzada

La validación cruzada es una técnica utilizada en la literatura para evaluar la eficacia de los modelos predictivos basados en ecuaciones de regresión lineal [46]. El sobreajuste es un problema común en la construcción de este tipo de modelos, y la validación cruzada ayuda a evitarlo mediante el remuestreo de datos. Durante la fase de construcción del modelo, la validación cruzada tiene por objeto estimar el rendimiento del modelo final cuando se trata con nuevos datos [47].

La validación cruzada aleatoria divide los conjuntos de datos en subconjuntos aleatorios en cada iteración. En cada iteración, se elige aleatoriamente una parte de los datos para entrenamiento y otra para validación. La partición de los datos puede variar cada vez que se ejecuta, lo que introduce variabilidad en los resultados. La validación cruzada aleatoria es útil en situaciones donde se quiere explorar diferentes particiones de los datos, especialmente en conjuntos de datos grandes, donde la validación cruzada tradicional puede ser computacionalmente costosa. Este método tiene la ventaja de que el número de iteraciones (k) es independiente del tamaño de los subbloques considerados. Sin embargo, la desventaja es que es imposible asegurar que todas las muestras formarán parte de los conjuntos de entrenamiento o validación, ya que muchas muestras pueden no ser elegidas nunca, y algunas pueden ser consideradas en más de una iteración, véase la Fig. 2.20.

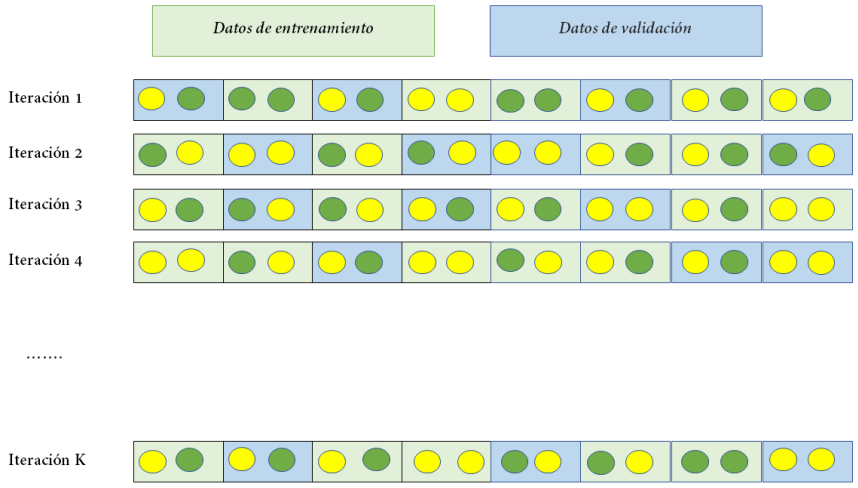


Figura 2.20: Validación cruzada aleatoria: En esta figura, los círculos verdes y amarillos representan las clases, mientras que los bloques verdes indican el conjunto de entrenamiento y los bloques azules corresponden al conjunto de validación.

2.4.13 Curvas ROC

La curva ROC (del inglés, Receiver Operating Characteristic) se presenta como un gráfico que muestra la sensibilidad de la prueba frente a los falsos positivos para distintos puntos de corte (el punto en que se intersecan dos puntos), los cuales son los más adecuados para una prueba determinada. También permite comprender su rendimiento general y comparar la capacidad diagnóstica de dos o más pruebas para distinguir entre dos resultados posibles. Al comparar distintos clasificadores, resulta útil disponer de una única medida que resuma el rendimiento de un clasificador. El área bajo la curva (AUC, por sus siglas en inglés) es una métrica importante que mide la capacidad de diferenciar clases, al calcular la probabilidad de clasificarlas correctamente. El método habitual de calcular el AUC se refiere a la probabilidad de que una instancia positiva (clase positiva) seleccionada al azar tenga una clasificación más elevada que una instancia negativa seleccionada al azar. En otras palabras, esto equivale al estadístico de suma de rangos de Wilcoxon de dos muestras. Es importante señalar que un clasificador con un AUC alto a veces puede tener peores resultados en una región específica que otro clasificador con un AUC más bajo. Sin embargo, en la práctica, el AUC es una medida fiable de la precisión con la que predice [48]. Los resultados pueden presentarse mediante la curva ROC véase Fig. 2.21 [49].

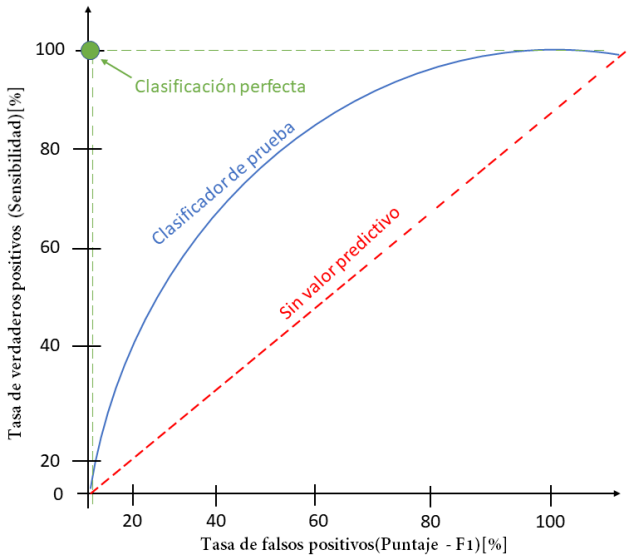


Figura 2.21: Casos de evaluación en la curva ROC: Las tres líneas representan: 1) la línea de no discriminación (diagonal) en color rojo, 2) la curva del modelo evaluado (clasificador de prueba) en color azul, y 3) la línea de clasificación ideal en color verde.

2.4.14 Metodología para DL

La metodología para DL se centra en la adquisición, procesamiento y evaluación de modelos de redes neuronales profundas, los cuales son capaces de aprender a distinguir entre una clase y otra a partir de los datos. A continuación, se presenta una metodología general para aplicar DL en este trabajo, relacionado con la clasificación de imágenes de nanomateriales.

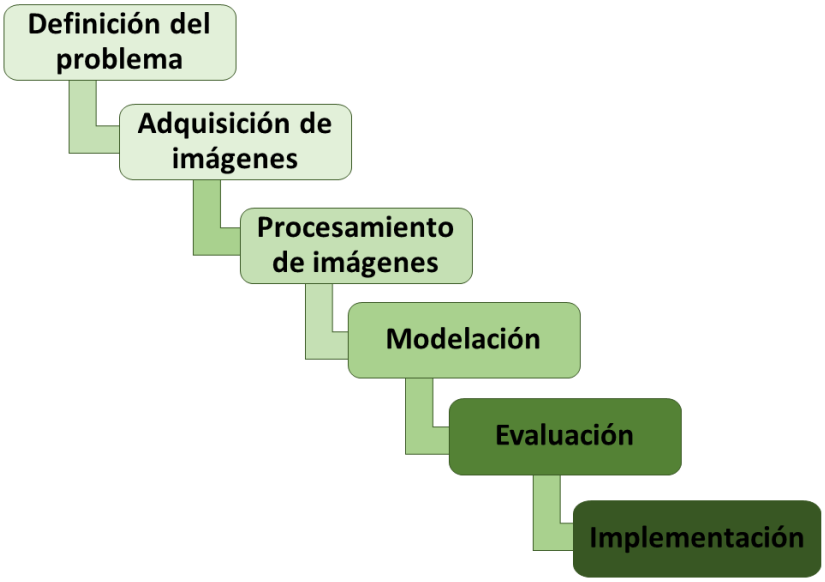


Figura 2.22: Metodología para DL.

Definición del problema

En esta etapa inicial, es importante que se definan los objetivos del proyecto de manera clara y que se comprendan de forma absoluta sus propósitos. Esto posibilita la transformación de dichos objetivos en un problema de clasificación de imágenes, mediante la elaboración de un plan preliminar que orienta las medidas necesarias para su consecución. Los proyectos de clasificación de imágenes, generalmente, se estructuran en función de las necesidades específicas de un área de aplicación, como la identificación de materiales o estructuras particulares. De este modo, se garantiza que los datos visuales recopilados, etiquetados y analizados sean relevantes y apropiados para el entrenamiento de modelos que respondan a los requisitos establecidos de clasificación.

Adquisición de imágenes

Este proceso consiste en la construcción de un conjunto de datos organizado por categorías, utilizando una o más fuentes de información. Estas fuentes pueden derivar de repositorios en línea o de laboratorios especializados en la recolección de muestras, asegurando la diversidad y calidad necesarias para cumplir con los objetivos del proyecto. En el caso de la clasificación de imágenes de nanomateriales, las imágenes se capturan utilizando técnicas avanzadas de microscopía electrónica, con el fin de obtener imágenes de alta resolución de los nanomateriales en cuestión. Las imágenes se organizan en diferentes categorías según el tipo de nanomaterial.

Procesamiento de los datos de imágenes

Esta etapa es crucial en el proceso de DL, ya que se realiza una revisión de las imágenes con el fin de asegurar su homogeneidad. El procesamiento de las imágenes se lleva a cabo mediante tareas como ajustar el tamaño, la forma y el color de las imágenes, para que puedan ser empleadas en las arquitecturas propuestas sin necesidad de operaciones inconsistentes.

Modelado

Tras procesar las imágenes, el siguiente paso es la identificación de un modelo que logre clasificar o etiquetar las imágenes con precisión. En esta etapa, los datos se dividen en dos subconjuntos clave: datos de entrenamiento y datos de validación. Los datos de entrenamiento se utilizan para enseñar al modelo a identificar patrones y características relevantes en las imágenes, mientras que los datos de validación se emplean para evaluar el rendimiento del modelo durante su entrenamiento, sin que este se ajuste excesivamente a los datos de entrenamiento, evitando así el sobreajuste. Este proceso permite determinar qué modelo es más eficiente, ya que proporciona una medida objetiva de su capacidad de generalización a datos no vistos. A través de métricas de desempeño, como la precisión, sensibilidad, el puntaje-F1 y la matriz de confusión, se puede comparar el rendimiento de varios modelos y ajustar sus parámetros para optimizar su efectividad en la clasificación de imágenes. En función de los resultados obtenidos, se selecciona el modelo más adecuado, el cual se entrenará de forma definitiva utilizando todos los datos disponibles.

Evaluación

En la etapa de evaluación, se mide el rendimiento del modelo utilizando datos no vistos durante el entrenamiento. Se calculan diversas métricas para determinar su efectividad y se analiza su capacidad para generalizar a nuevos datos. También se ajustan parámetros y se realizan pruebas adicionales para asegurar que el modelo sea adecuado y preciso antes de su implementación final.

Implementación

En la etapa de implementación, el modelo evaluado se pone en uso en un entorno real o productivo. Esto incluye integrar el modelo en el sistema o aplicación que lo utilizará, asegurando que pueda procesar nuevos datos de manera eficiente y precisa. Además, se monitorea su desempeño de forma continua para detectar posibles desviaciones o fallos y, en caso necesario, realizar ajustes o actualizaciones. Esta etapa también puede implicar la optimización del modelo para mejorar su velocidad, escalabilidad y capacidad para manejar grandes volúmenes de datos, garantizando su funcionalidad a largo plazo.

3. Estado del arte

Esta sección de la investigación presenta una revisión de la literatura, en la que se abordan diversas contribuciones realizadas en el área de la clasificación de imágenes utilizando aprendizaje profundo. El análisis de estos trabajos se ha dividido en cuatro categorías: (i) clasificación y aprendizaje por transferencia, (ii) clasificación y detección de imágenes mediante aprendizaje profundo, (iii) clasificación basada en aprendizaje automático, y (iv) optimización de hiperparámetros.

Las bases de datos consultadas para esta revisión fueron *Google Scholar*, así como *Scopus* e *IEEE Xplore*. Aunque *Google Scholar* incluye trabajos provenientes de múltiples bibliotecas y repositorios, también se realizaron búsquedas adicionales en *Scopus* e *IEEE Xplore* para garantizar la inclusión de estudios que, aunque no están disponibles en acceso abierto, podrían no haber sido indexados en otras plataformas especializadas debido a restricciones de pago. La recopilación de los artículos para esta revisión se centró principalmente en la clasificación de imágenes mediante aprendizaje profundo, aunque también se incluyeron algunos relacionados con el aprendizaje automático y la optimización de hiperparámetros, debido a su relación indirecta.

Los artículos fueron evaluados en función del título, las palabras clave y el resumen, asegurándose de que las palabras clave empleadas en cada artículo coincidieran con el tema de este trabajo, con el fin de garantizar la pertinencia de los estudios seleccionados.

3.1 Clasificación y aprendizaje por transferencia

En [50] se presenta un enfoque de aprendizaje por transferencia para la clasificación de nanomateriales de carbono a partir de imágenes de microscopio electrónico de transmisión (TEM, por su sigla en inglés). Debido a la falta de herramientas automatizadas, la clasificación de imágenes de estas nanoestructuras se realiza manualmente. Para automatizar este proceso, se propone el uso de técnicas

de visión por computadora y ML basado en CNN. Utilizando un conjunto de datos compuesto por 5323 imágenes en escala de grises de nanoestructuras suspendidas en el aire, se aplicaron los métodos K-means y TL para generar vectores de hipercolumnas, los cuales fueron agrupados mediante K-means. Se utilizó una función de activación softmax con un algoritmo de aumento de gradiente, empleando el método de Vector de Descriptores Localmente Agregados. El modelo alcanzó una precisión del 90.9% al clasificar 4 clases y del 84.5% con 8 clases, demostrando su capacidad para detectar y clasificar automáticamente las nanoestructuras. Este trabajo tiene aplicaciones potenciales en la clasificación de nanoestructuras.

Respecto a [51], se emplearon modelos de DL en CNN y TL, utilizando el modelo Inception de Google para entrenar y clasificar las imágenes. La investigación fue motivada por la necesidad de mejorar el reconocimiento de objetos en imágenes de alta resolución, un área clave en la clasificación. Aunque se han desarrollado varios modelos de clasificación, aún persisten problemas técnicos por resolver. Para llevar a cabo la investigación, se utilizaron imágenes de Google, con las que se formo una base de datos compuesta por 1369 imágenes de cinco tipos de flores: margaritas, dientes de león, rosas, girasoles y hojas de tulipán. Los resultados demostraron que la combinación de ambos enfoques permitió alcanzar una precisión del 88.3% en el conjunto de prueba, mientras que el modelo sin TL solo logró un 60.2%. Este hallazgo aporta el conocimiento de que la integración de DL con TL mejora significativamente la precisión en la clasificación de imágenes.

En cuanto a [52], este estudio resalta la eficacia de las CNN, como AlexNet, VGG16 y VGG19, en la extracción robusta de características para tareas de reconocimiento y segmentación de objetos. Utilizando las bases de datos CalTech256 [53] y GHIM10K [54], se comparó el desempeño de VGG19 frente a AlexNet, VGG16 y un enfoque híbrido que combina CNN con un clasificador de máquinas de vectores de soporte. La experimentación incluyó dos etapas: entrenamiento y prueba de las CNN, empleando métricas como exactitud, sensibilidad y puntaje-F1 para evaluar el rendimiento. VGG19 obtuvo los mejores resultados, superando a las demás arquitecturas y al modelo híbrido. El estudio sugiere que VGG19 podría ser útil en aplicaciones como la detección de objetos y el reconocimiento de acciones humanas.

3.2 Clasificación y detección de imágenes mediante aprendizaje profundo

Aplicaciones como la presentada en [55] resaltan la importancia del análisis fractográfico para optimizar el rendimiento y la seguridad de piezas mecánicas,

identificando texturas y marcas superficiales para determinar el tipo y la causa de fallas. El estudio utilizó dos conjuntos de datos: 858 imágenes a escala real y 608 imágenes obtenidas mediante microscopía electrónica de barrido a diferentes escalas. Se compararon dos enfoques para clasificar fracturas: CNN preentrenadas y redes wavelet adaptativas profundas (DAWN, por sus siglas en inglés). Ambos lograron resultados similares, pero DAWN destacó por su menor uso de parámetros, consolidándose como un enfoque innovador basado en análisis multirresolución.

Bajo el enfoque de [56] se desarrolló un sistema basado en DNN para clasificar dos familias de macroinvertebrados: Calopterygidae y Heptageniidae. Se creó una base de datos organizada en carpetas por especie, con 200 imágenes seleccionadas aleatoriamente para cada categoría, distribuidas en un 70% para entrenamiento y un 30% para validación. La tarea de clasificación se realizó utilizando la red preentrenada Inception-v3, implementada en Python con las bibliotecas TensorFlow y Keras, y complementada con un análisis de variantes mediante la herramienta ML Works. Los resultados mostraron una precisión del 90% en el conjunto de entrenamiento, destacando la efectividad del modelo en la clasificación de estas especies.

Por su parte, [57] presenta una revisión de varios trabajos en los que se aborda el enfoque práctico del DL en la mejora de la microscopía electrónica. Como ventaja, se resalta que no es necesario que los expertos elaboren ecuaciones complejas, ya que el DL puede generarlas automáticamente, lo que permite a los investigadores ser más eficientes. Además, gracias a los avances en el aprovechamiento computacional, los procesos pueden completarse en segundos. Sin embargo, Como desventaja, se menciona que el DL no aprovecha los conocimientos especializados de la física, específicamente los que se utilizan para optimizar los cálculos de información entre los usuario. La revisión de los trabajos ofrece una visión general de las aplicaciones actuales del DL en la microscopía electrónica, como, por ejemplo, el etiquetado y la segmentación semántica. La investigación se llevó a cabo mediante un análisis exhaustivo de las aplicaciones del DL en este campo, abordando los requerimientos de hardware y software necesarios para implementar esta tecnología en la microscopía electrónica, así como la interfaz entre los microscopios electrónicos y las herramientas de DL. Además, se revisaron los componentes fundamentales de las redes neuronales, las arquitecturas populares y los métodos de optimización. El conocimiento aportado por este estudio facilita una comprensión más clara de las herramientas y técnicas clave para aplicar el DL en la microscopía electrónica, lo que abre el camino para posibles mejoras en este campo mediante el uso del DL.

El artículo [58] presenta un sistema innovador para el análisis de bacterias mediante redes neuronales profundas, utilizando imágenes obtenidas mediante

SEM. Este enfoque responde a la necesidad de mejorar la caracterización de las bacterias intestinales, un factor crucial para evaluar el estado de salud de una persona. El sistema emplea una red U-Net preentrenada para procesar las imágenes. La investigación se estructura en tres etapas principales: segmentación, separación y clasificación. Para ello, se utilizaron tres conjuntos de datos denominados Bacteria Set 1, Bacteria Set 2 y Bacteria Set 3, con 100 imágenes en cada conjunto, para un total de 300 imágenes en escala de grises, originalmente de 960 × 1280 píxeles, redimensionadas a 400 x 400 píxeles. Durante la clasificación se aplicaron técnicas de aumento de datos para optimizar la orientación de las imágenes. Los resultados obtenidos mostraron una precisión de sensibilidad total del 99,2 %, 95,8 % y 90,1 % en los conjuntos de datos respectivos. Este estudio presenta un sistema que no solo mejora la precisión en el conteo y cálculo de la proporción bacteriana, sino que también facilita la identificación de regiones bacterianas, la separación de áreas conectadas y su clasificación. Esto representa un avance significativo en comparación con los métodos convencionales, con aplicaciones potenciales en la evaluación de la salud intestinal y el análisis microbiológico.

En [59], se implementó una técnica basada en CNN para clasificar NPs en cuatro clases de diámetros (40, 50, 60 y 80 nm), utilizando un análisis detallado de mapas de dispersión bidimensional generados a partir de experimentos en superficies de NPs. El estudio identificó que la generación de estos mapas es un proceso lento y exigente en términos de tiempo. Además, se llevó a cabo un análisis de circuitos integrados en la industria de semiconductores, centrado en la calidad de las obleas utilizadas en máquinas de litografía. Se descubrió que partículas con diámetros entre 1 micrómetro y 20 nanómetros pueden afectar significativamente la calidad de estas obleas. Para mejorar la detección e inspección no invasiva de superficies de NPs, se recomendó el uso del método de dispersión Fourier coherente (CFS, por sus siglas en inglés: Coherent Fourier Scattering Method), una técnica empleada en el análisis de materiales y estructuras, que asegura un alto rendimiento en términos de calidad. En este estudio, se utilizó una base de datos de 1302 imágenes. Este enfoque superó a métodos tradicionales basados en umbralización y búsqueda, logrando un incremento del 2 % en precisión para partículas con diámetros de 40 y 50 nm. Las CNN mostraron ventajas significativas al extraer características de manera automática, mejorando la separación de NPs con patrones de dispersión colaterales similares entre clases. El autor concluyó que las CNN son herramientas clave para la detección y clasificación eficiente de NPs, especialmente cuando se enfrentan a imágenes nunca antes vistas por el modelo.

En [60], se aborda un desafío común en las imágenes SEM: la aparición de imágenes borrosas causadas por desajustes en el hardware y errores en la automatización. La restauración de estas imágenes es crucial para obtener

resultados precisos en el análisis microestructural, especialmente en la ciencia de materiales. Para resolver este problema, se propuso un método basado en DL para reenfocar imágenes de SE, con el objetivo de mejorar la eficiencia en la adquisición y análisis de imágenes SEM. Se implementaron tres enfoques basados en CNN: una CNN de escala única, una CNN de múltiples escalas y una versión mejorada de esta última, impulsada por aumento de datos. Este enfoque progresivo permitió evaluar y comparar la capacidad de restauración de imágenes SEM borrosas en distintas configuraciones y niveles de complejidad. Los resultados demostraron que el método propuesto no solo es capaz de reenfocar imágenes SEM de baja calidad, sino que también puede hacerlo de manera discriminatoria, mejorando específicamente las regiones desenfocadas sin afectar el resto de la imagen. El estudio se basó en un conjunto de 33 pares de imágenes (una enfocada y una desenfocada) en escala de grises, con dimensiones originales de 2048 x 1536 píxeles. Estas imágenes fueron recortadas a 256 x 256 píxeles y ampliadas a 1584 pares, que fueron divididos en subconjuntos para el entrenamiento (80%), validación (10%) y prueba (10%) del modelo. Los enfoques propuestos demostraron un rendimiento sólido tanto en la restauración de imágenes SEM de acero martensítico como en aleaciones endurecidas por precipitación, lo que demostró la efectividad de la solución en diferentes tipos de materiales. El aporte de este estudio es significativo, ya que ofrece una solución efectiva y automatizada para la restauración de imágenes SEM borrosas, lo que acelera la adquisición de imágenes de alta calidad y optimiza el tiempo de procesamiento.

En [61], se presenta un método basado en DL aplicado a la caracterización de materiales mediante imágenes SEM, utilizando la difracción de electrones retrodispersados (EBSD, por sus siglas en inglés). Este enfoque integra técnicas de clasificación y cuantificación, empleando una red neuronal preentrenada U-Net que establece correlaciones entre las imágenes SEM y los datos de microestructura de aceros de ingeniería. La red utiliza capas de omisión, características propias de U-Net, lo que mejora el análisis guiado y refuerza la precisión del método EBSD, considerándolo como base esencial para el buen desempeño del modelo. Este avance también propone un concepto innovador para la implementación de métodos híbridos de DL en metalurgia física, favoreciendo el desarrollo de la metalografía cuantitativa de alto rendimiento y acelerando la optimización de materiales. El método propuesto utiliza mapas EBSD para definir la verdad fundamental de las imágenes SEM y aplica una red U-Net para realizar una segmentación píxel por píxel. A través del análisis de píxeles con OpenCV [62], se obtiene información microestructural cuantitativa. Este enfoque mejora la precisión del análisis de la microestructura, con aplicaciones en estudios de materiales y optimización metalúrgica.

Por su parte, en [63] se presenta el estudio de una DNN supervisada para

eliminar el ruido presente en imágenes TEM de NPs de platino. En este estudio se revisan diversos métodos para la eliminación de ruido, utilizando una base de datos de 17,955 imágenes con una resolución de 1024 × 1024 píxeles a nivel atómico. Para ello, se emplearon aproximadamente 5,500 imágenes simuladas para entrenar la red, y se seleccionaron al azar otras 550 imágenes para su validación y prueba. Estas imágenes fueron agrupadas mediante interpolación cúbica con multicortes. Se probaron métodos de eliminación de ruido en tamaños de 400 × 400 píxeles, 900 × 900 píxeles, 600 × 600 píxeles y 128 × 128 píxeles. Los métodos evaluados incluyeron una versión modificada de U-Net, eliminación de ruido de puntos basada en redes U-Net ciegas, y un estimador de riesgo imparcial de Poisson con expansión lineal de umbrales, posteriormente. Como resultado, se lograron imágenes limpias y libres de ruido, mejorando significativamente la calidad de las imágenes de alta resolución. Este enfoque resulta particularmente útil en investigaciones que requieren un análisis preciso de imágenes TEM.

En [64], se propone un sistema para la detección de anomalías en imágenes de SEM, basado en CNN y análisis de similitudes. Este enfoque evalúa las anomalías por región, calculando similitudes en función de parámetros definidos, y destaca la utilidad de métodos basados en similitudes visuales para describir el contenido de imágenes mediante redes neuronales. Los autores sugieren que este enfoque puede extenderse a otras áreas de microscopía, como TEM. El sistema fue probado con imágenes de nanofibras y demostró resultados satisfactorios. Se desarrollaron dos variantes del método, evaluadas con un conjunto de datos público de imágenes SEM. Ambas superaron el rendimiento de técnicas existentes en aproximadamente un 5%, alcanzando un AUC cercana al 97%. Las variantes procesaron imágenes SEM y TEM de 700 x 1024 píxeles en un ordenador personal, requiriendo aproximadamente 50 y 15 segundos, respectivamente. Los resultados experimentales muestran que el método detecta con alta precisión defectos de grano grueso, aunque es menos eficaz para defectos de grano medio y fino, lo que destaca una oportunidad para futuras mejoras.

3.3 Clasificación basada en aprendizaje automático

De acuerdo con un análisis realizado en [65], se pueden presentar defectos durante el proceso de hilado de materiales nanofibrosos, lo cual puede generar dificultades de producción en un entorno industrial. En consecuencia, la investigación y la supervisión de anomalías ha adquirido un papel esencial, y se han desarrollado sistemas inteligentes con el fin de brindar soluciones efectivas. Los autores plantean un enfoque basado en el ML para la detección automática de anomalías, utilizando

una CNN para analizar imágenes SEM de dos tipos de nanofibras: nanofibras homogéneas (HNF, por sus siglas en inglés) y nanofibras no homogéneas (NHNF, por sus siglas en inglés). Se recolectó una base de datos manualmente con un total de 160 imágenes de las cuales 85 correspondían a NHNF y 75 imágenes HNF, de las cuales 70% de los datos se destinaron a entrenamiento y 30% a validación de nanofibras de polivinilacetato. Los resultados mostraron que la clasificación con CNN alcanzó un 80% de precisión.

Respecto a [66], se identificó la necesidad de realizar una revisión precisa de las imágenes SEM en la fabricación de nanomateriales nanofibrosos producidos por hilado electrostático. Esta necesidad surge debido a que los defectos estructurales y las anomalías pueden dificultar el uso de estos materiales en aplicaciones dentro del campo de la nanotecnología. Para abordar este desafío, se propuso un sistema de clasificación automática que evita el procesamiento innecesario de imágenes. El sistema se basa en técnicas de ML tanto supervisadas como no supervisadas, combinando un codificador automático (AE, por sus siglas en inglés) con aprendizaje no supervisado y un perceptrón multicapa (MLP, por sus siglas en inglés), también con aprendizaje no supervisado. El objetivo del sistema es clasificar las nanofibras en dos categorías: con y sin anomalías, generando vectores distintivos para cada imagen de entrada. Las técnicas híbridas AE-MLP demostraron un rendimiento superior, alcanzando una precisión del 92 %, en comparación con métodos tradicionales de ML. No obstante, el sistema aún presenta limitaciones, como la necesidad de que un experto seleccione manualmente las regiones de la imagen para su análisis.

3.4 Enfoque de configuración mediante optimización de hiperparámetros

Es esencial realizar un análisis exhaustivo que permita prevenir fallos, especialmente cuando se trata de la optimización de los hiperparámetros. La configuración de un experimento depende de diversos factores, algunos de los cuales pueden ajustarse mediante la optimización de hiperparámetros, mientras que otros permanecen constantes. En los próximos artículos, se analizarán investigaciones que abordan estos factores clave, centrándose en cómo la optimización de hiperparámetros puede influir en el rendimiento de los modelos. Se buscará aplicar las conclusiones y recomendaciones de estos estudios en futuras investigaciones, con el objetivo de mejorar la precisión y eficiencia en la configuración experimental.

En [67], los autores abordan los desafíos que surgen al realizar experimentos en el ámbito del DL, como el sobreajuste y la alta demanda de tiempo de ejecución. Para abordar estos problemas, llevaron a cabo un estudio empírico en el que

implementaron capas de dropout y normalización por lotes en redes neuronales densas multicapa y en CNN. El estudio destaca la importancia de evaluar el rendimiento de los modelos de DL en función de varios factores, como el tiempo de CPU durante el entrenamiento y la prueba, el tamaño del modelo (número de parámetros) y la precisión de clasificación. Se observó que un optimizador no adaptativo, como el descenso de gradiente estocástico, puede ofrecer un mejor rendimiento que los optimizadores adaptativos, aunque a costa de un mayor tiempo de entrenamiento para ajustar los hiperparámetros. En contraste, los optimizadores adaptativos, como RMSProp, requieren menos ajuste y funcionan bien de forma predeterminada. La investigación también concluyó que tanto la normalización por lotes como las capas dropout deben ser utilizadas con precaución en las CNN, recomendando experimentar con estas técnicas. Si se dispone de tiempo limitado o hay incertidumbre, se sugiere priorizar la normalización por lotes. En general, la interacción entre la estructura de la red, las capas dropout y la normalización por lotes proporciona información valiosa sobre cómo y cuándo aplicar estas técnicas para mejorar el rendimiento en el DL.

En otro caso, en [68], se analiza el uso de las CNN en la clasificación de imágenes, destacando que, aunque son la técnica más utilizada, la elección de la arquitectura óptima sigue siendo un reto, ya que el rendimiento de las CNN depende de múltiples hiperparámetros. El artículo introduce un nuevo método denominado Propagación de Modelos CNN de Élite Mejorada (E-Enhanced CNN-MP, por sus siglas en inglés), que permite aprender automáticamente la estructura óptima de una CNN. Para ello, se utilizó la base de datos Caltech-256, dividiendo el 70 % para entrenamiento y el 30 % para prueba. Este enfoque emplea AG, conocidos por su capacidad para resolver problemas complejos, como identificar los mejores hiperparámetros para una tarea de clasificación específica. Las simulaciones realizadas por los autores demuestran la eficacia del método, alcanzando una precisión del 98.94 % al determinar los hiperparámetros óptimos para la tarea de clasificación. Además, se presenta la estructura de la CNN obtenida utilizando el método E-Enhanced CNN-MP mejorado, que demuestra un alto rendimiento en la optimización de la red.

En [69], el autor se enfoca en la optimización de los hiperparámetros de modelos de ML, un paso crucial en la aplicación de cualquier algoritmo. Este proceso implica ajustar los valores de los hiperparámetros para minimizar los errores en las pruebas. En el caso de los modelos de DL, la optimización de hiperparámetros no solo abarca los parámetros del modelo, sino también su arquitectura. Esto incluye la selección de parámetros como la tasa de aprendizaje y el tamaño del lote, así como la determinación de la estructura del modelo, como el número de capas y neuronas en cada capa. El artículo presenta diversas pruebas utilizando métodos de optimización, tales como la búsqueda de cuadrilla, la búsqueda aleatoria y el

estimador de Parzen estructurado en árbol (TPE, por sus siglas en inglés), para mejorar la selección de estos hiperparámetros y así optimizar el rendimiento de los modelos de DL.

Por otra parte, en [70] se aborda el TL, destacando que la práctica común consiste en ajustar los hiperparámetros de un modelo preentrenado, tales como la tasa de aprendizaje y el tamaño del lote, basándose en los valores utilizados en el entrenamiento desde cero, como en el caso de ImageNet. Sin embargo, el ajuste fino de estos hiperparámetros ha sido un campo poco explorado, ya que la mayoría de los estudios previos se han centrado en ajustar parámetros de referencia en evaluaciones centradas en el TL. El experimento del autor se enfoca en determinar los hiperparámetros óptimos, ajustando la tasa de aprendizaje efectiva y teniendo en cuenta las diferencias en la procedencia de los conjuntos de datos entre clases.

En [71], se compara el impacto de las funciones de activación en el aprendizaje automático y el aprendizaje profundo. El estudio subraya la importancia de estas funciones en el rendimiento de las redes neuronales, presentando una comparación entre la función sigmoidea y ReLU. Además, el artículo explora las ventajas de las redes neuronales profundas y su dependencia de los pesos, destacando cómo estas funciones de activación influyen en la capacidad de aprendizaje y en la eficiencia de las redes neuronales.

Bajo el enfoque de [72], los autores proponen modificaciones en los parámetros de las arquitecturas existentes para mejorar el rendimiento de las CNN en el reconocimiento de imágenes, una tarea ampliamente utilizada. En este estudio, se desarrolla un algoritmo basado en la optimización de enjambre simplificado para optimizar los hiperparámetros de una de las arquitecturas más simples de CNN, LeNet. Los parámetros ajustados incluyen la tasa de aprendizaje, el número de épocas y el optimizador, con el objetivo de mejorar la eficiencia del modelo.

En [73], se destaca que, aunque la implementación y el uso de las CNN se han simplificado con los avances tecnológicos, encontrar la arquitectura y los hiperparámetros óptimos para cada tarea sigue siendo crucial para obtener los mejores resultados. Dado el gran número de hiperparámetros involucrados, identificar la configuración ideal puede ser una tarea compleja, sin un enfoque determinista. En las etapas iniciales del desarrollo de las CNN, el método más utilizado para ajustarlas es el de conjeturas y estimaciones, también conocido como el método de estimación del anfitrión. Para problemas de optimización más difíciles, los algoritmos de inteligencia de enjambre emergen como una solución prometedora.

De acuerdo con [74], se comparó el rendimiento de dos arquitecturas de CNN: AlexNet e Inception-v3. Se descubrió que al utilizar TL con la red Inception-v3, se alcanzó una precisión máxima del 94.11%. Para optimizar el rendimiento, se empleó el optimizador Adam con una tasa de aprendizaje de 0.0001, un tamaño

de lote de 256 y se entrenó el modelo durante 13 épocas.

En [75], se aborda la optimización de hiperparámetros como un paso crucial para lograr una alta precisión con las CNN. Determinar los valores exactos de los hiperparámetros para un rendimiento óptimo es un reto, ya que varía según el conjunto de datos utilizado. Debido a la amplia gama de valores posibles y al gran número de combinaciones, este proceso puede resultar en elevados costes computacionales. Para abordar este desafío, el autor empleó la búsqueda de cuadrilla, lo que permitió reducir el rango de valores de los hiperparámetros y optimizar el proceso de búsqueda. Los resultados experimentales confirmaron que la tasa de aprendizaje es el hiperparámetro que más influye en el rendimiento del modelo, y que al reducir el rango de búsqueda y aplicar la búsqueda de cuadrilla, se puede mejorar la precisión y reducir eficazmente los cálculos necesarios.
Por parte de [76], en los sistemas de ML existen muchos hiperparámetros, como la tasa de aprendizaje, que afectan las predicciones finales. Sin embargo, encontrar una buena combinación de hiperparámetros puede suponer un reto, ya que su evaluación requiere una potencia computacional significativa. El trabajo del autor pretende resolver este problema prediciendo la mejor combinación de hiperparámetros sin completar el costoso proceso de aprendizaje. Para ello se recopilan datos sobre el rendimiento del modelo y se utilizan para determinar qué combinaciones de hiperparámetros son las más prometedoras. Los experimentos preparados mostraron que este enfoque mejora el método de búsqueda aleatoria utilizado habitualmente.

En cambio, en [77], se señala que los algoritmos de ML se ven influenciados por diversos factores que pueden afectar su rendimiento. Entre estos factores se encuentran la cantidad, la calidad y las características de los datos. Al elegir un algoritmo adecuado, es esencial considerar otros elementos, como la configuración de los parámetros, que deben basarse en la especificación del problema. Existen dos tipos de parámetros configurables: los internos del modelo y los externos. Los parámetros internos del modelo pueden estimarse a partir de los datos, utilizando los pesos de una DNN. En cambio, los parámetros externos, como la tasa de aprendizaje para entrenar una red neuronal, no pueden estimarse a partir de los datos. Los hiperparámetros pueden fijarse por expertos o ser obtenidos mediante una búsqueda de parámetros o a partir de otros problemas. Sin embargo, los mejores valores de estos parámetros se encuentran ajustando el algoritmo para lograr la mayor precisión. Este trabajo tiene como objetivo comparar distintos algoritmos de optimización para identificar los mejores valores de hiperparámetros para la red neuronal. Para ello, se aplicaron algoritmos de búsqueda en cuadrilla, bayesiano y AG, los cuales fueron evaluados utilizando medidas como la precisión y el tiempo de ejecución.

Por ejemplo, [78] se centra durante la pandemia de COVID-19 en 2020,

algunos investigadores estudiaron la frecuencia de la actividad del tacto facial en la vida cotidiana. Para construir un sistema que evite el tacto facial, se han propuesto algoritmos basados en DL que han mostrado resultados prometedores. Sin embargo, uno de los principales retos del DL es encontrar los hiperparámetros adecuados, como el tamaño del filtro, el número de filtros, el tamaño del lote, el número de épocas y la técnica de optimización del entrenamiento. Estos hiperparámetros pueden afectar en gran medida el rendimiento del algoritmo. En este estudio, se presentó un enfoque eficiente para el ajuste de hiperparámetros de CNN. El objetivo era reconocer actividades de tacto facial utilizando datos de acelerómetros. Se evaluaron dos métodos de ajuste de hiperparámetros: la búsqueda de cuadrilla y la optimización bayesiana. Los resultados mostraron que la optimización bayesiana proporcionaba los mejores hiperparámetros para las CNN, con una precisión del 96.61% para el reconocimiento de contactos faciales.

Por otra parte, en [79] se abordó el problema del sobreajuste mediante tres experimentos, en los cuales se probaron diferentes combinaciones de hiperparámetros dentro de la arquitectura Faster RCNN. Los hiperparámetros clave ajustados para mejorar el rendimiento del modelo fueron la inicialización de pesos y el optimizador. Los resultados fueron notables, logrando mejoras significativas en comparación con trabajos previos. El modelo mejorado de detección de objetos alcanzó una tasa de sensibilidad del 93.94%, con una puntuación mínima de verdaderos positivos del 98%.

En [80] se describe que las CNN no sólo son útiles para el reconocimiento de imágenes, sino también para mejorar el rendimiento de los modelos CNN en muchos campos. La optimización de los hiperparámetros de los modelos CNN es esencial para mejorar su rendimiento. En este trabajo se propone un método basado en AG para optimizar los hiperparámetros de los modelos CNN que se utilizan para clasificar los datos que se encuentran en la base de datos MNIST[81]. A diferencia de estudios anteriores, los algoritmos basados en poblaciones pueden utilizarse para optimizar simultáneamente múltiples parámetros. Además, en este método se utilizan diferentes tipos y rangos de parámetros de AG. Los resultados del estudio presentan los valores de hiperparámetro que logran la mejor precisión de clasificación para los datos MNIST.

También, en [82] se señala que las CNN enfrentan un reto importante en el diseño de su arquitectura, una tarea que requiere una considerable cantidad de tiempo, recursos y conocimientos. Esto se debe al gran número de opciones de prueba en los parámetros de las CNN, considerando todos los aspectos que deben ser evaluados. Hasta ahora, los humanos han sido los principales responsables de crear estas arquitecturas, lo que puede resultar una tarea desalentadora. En este trabajo, se analizaron varios enfoques recientes, como la búsqueda de arquitecturas neuronales (NAS, por sus siglas en inglés), junto con sus limitaciones, y se propuso un nuevo método para llevar a cabo este proceso. El modelo propuesto utiliza un espacio de búsqueda simplificado, con una estrategia de búsqueda de cuadrícula aleatoria. Además, se empleó un método de búsqueda de arquitectura basado en celdas, donde una celda comprende múltiples operaciones CNN, junto con varias opciones de enlace dentro de los nodos de operación de la celda. El modelo propuesto se probó en el conjunto de datos MNIST, demostrando un rendimiento significativo y comparable con las arquitecturas de última generación para MNIST. La base de datos MNIST contiene 60000 imágenes de entrenamiento y 12000 imágenes de prueba. Se entrenaron 93 arquitecturas con 5 épocas. De estas 93 arquitecturas, se eligieron las 5 mejores para un entrenamiento completo con aumento de datos. Se obtuvo una precisión del 99.05 %, y la mejor precisión en MNIST es del 99.63 %, obtenida por una de las 5 mejores arquitecturas elegidas.

En [83] se describen diversas aplicaciones del mundo real que requieren optimización en situaciones dinámicas, donde el reto consiste en localizar y seguir los óptimos de una función objetivo dependiente del tiempo. Para abordar los problemas de optimización dinámica (DOP, por sus siglas en inglés), se han desarrollado varias técnicas evolutivas. Sin embargo, aún se necesitan soluciones más eficientes. Recientemente, ha surgido una nueva tendencia en la optimización en entornos dinámicos, en la que se espera que los nuevos algoritmos de aprendizaje por refuerzo (RL, por sus siglas en inglés) mejoren los métodos utilizados en la comunidad DOP. Este trabajo propone un algoritmo de optimización basado en RL y Q-learning, denominado ROA (por sus siglas en inglés), para la optimización de hiperparámetros de redes neuronales convolucionales (CNN). Se utilizaron dos conjuntos de datos, el conjunto de datos MNIST y el conjunto de datos CIFAR-10 [84], para probar el modelo de RL propuesto. Debido al uso de RL para la optimización de hiperparámetros, la CNN optimizada con ROA produjo resultados muy competitivos y mostró un buen rendimiento. Los resultados experimentales indican que la CNN optimizada con ROA presenta una mayor precisión que la CNN sin optimización. Cuando se utiliza el conjunto de datos MNIST, la precisión de la CNN optimizada con ROA, tras entrenar durante 5 épocas, fue del 98.97%, superior al 97.62% obtenido por la CNN sin optimización. En el caso del conjunto de datos CIFAR-10, la precisión de la CNN optimizada con ROA, tras entrenar durante 10 épocas, alcanzó el 73.40%, superando el 71.73% de la CNN sin optimización.

En este trabajo, hemos considerado varios artículos utilizados en diferentes áreas con fines de evaluación. Estos artículos muestran similitudes en la clasificación, por lo que los hemos tenido en cuenta en nuestra discusión y experimentación. Además, hemos recopilado una lista de hiperparámetros que se modificaron y la presentamos en la tabla de la Fig. 3.1.

Título	*Año*	*Tasa de aprendizaje*	*Minilotes*	*Épocas*	*Capas*	*Optimizadores*
Simplified swarm optimization for the hyperparameters of a convolutional neural network.	2023	0.4,0.7,0.9	128	10	...	SGD
Convolutional neural networks hiperparameters tuning.	2021	...	10-100	..	..	Adam, SGD
Convolutional neural network hiperparameter tuning for classifying firearms on images.	2022	0.01,0.001,0.0001	32,64,128	10,200	..	Adadelta, adagrad,Adam, RMSprop y SGD
Hyperparameter optimization for image classification in convolutional neural network.	2020	0.001,0.1	64	..	3,6	..

Sigue en la página siguiente.

Título	*Año*	*Tasa de aprendizaje*	*Minilotes*	*Épocas*	*Capas*	*Optimizadores*
Hyperparameter optimization; comparing genetic algorittm against Grid search and Bayesian optimization.	2021	..	..	..	1,3	Adam, RMSprop y Bynary cross entropy
Hyperparameter Tuning in convolutional neural network for face touching activity recognition using accelerometer date.	2022	..	64	200	..	Adam
Hyperparameters Tuning off aster R-CNN Deep learning Transfer For persistent Object detection in radar images.	2022	0.5 – 0.9	1,2,4	2000 000, 5000	..	SGDM, Adam

Sigue en la página siguiente.

Título	***Año***	***Tasa de aprendizaje***	***Minilotes***	***Épocas***	***Capas***	***Optimizadores***
Optimización of hyperparameter for CNN model using genetic algorithm.	2019	0.1, 0.0001	50	..	4	..
Randomized search on a grid of CNN networks with simplified search space.	2021	..	..	..	..	Adam
RL based hyperparameters optimization algorithm (ROA) for convolutional neural network.	2022	0.001, 0.003, 0.01, 0.03	..	5, 10	..	..

Tabla 3.1: Hiperparametros usados en trabajos de investigación..

4. Materiales y métodos

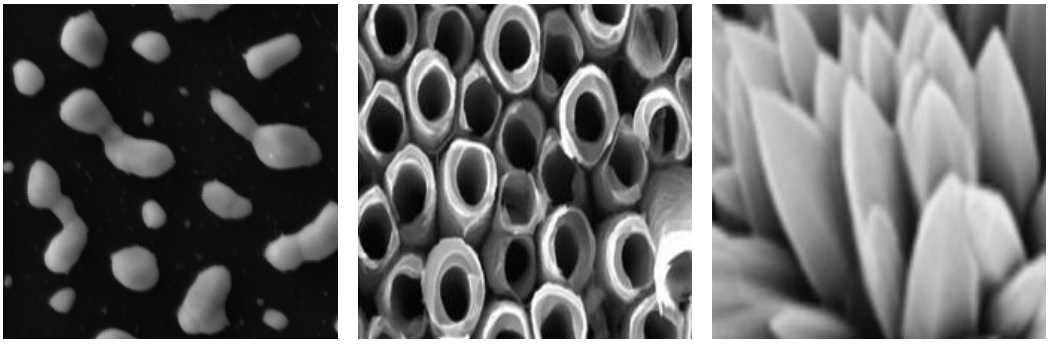
En este capítulo se abordan los materiales utilizados y los métodos seguidos en este trabajo de investigación. Primero, se presenta una descripción detallada de los datos utilizados; posteriormente, se da a conocer el software y hardware empleados en los experimentos; y por último, se explican detalladamente las diferentes etapas de la metodología.

4.1 Descripción de los datos

El conjunto de datos incluye 240 imágenes, divididas en tres categorías: NPs, NTs y NSs; con 80 imágenes en cada categoría, cuyos materiales corresponden a plata, carbono y paladio. Una parte de las imágenes se obtuvieron del Laboratorio de Semiconductores, Microelectrónica y Nanotecnología del Instituto de Ingeniería de la Universidad Autónoma de Baja California, utilizando un SEM modelo LYRA3 de la marca TESCAN. Las imágenes se capturaron utilizando los siguientes parámetros: detector SE, con un voltaje de aceleración de 5-20 kV, magnificación de 2.5-225 kx y distancia de trabajo de 9.0 mm. Otra parte de las imágenes fueron obtenidas de diversos repositorios en línea capturadas con distintos microscopios y parámetros [85]. Finalmente, todas las imágenes recolectadas tuvieron formatos jpg y tif, así como resoluciones de 768 x 858, tamaño de 16KB a 1MB y profundidad de 8bits, véase muestras de imágenes de base de datos en la Fig. 4.1.

4.2 Software y hardware

Para llevar a cabo los experimentos, se utilizaron dos ordenadores, uno personal y otro en un servicio de computación en la nube proporcionado por Amazon Web Services. Se utilizó el software de programación MATLAB R2023b para ejecutar los códigos en un ordenador de escritorio con una CPU Intel Core i7-13700k a un procesador de 3.40 GHz, 32 GB de RAM, gráficos RTX 3060 Ti y funcionando con



(a) NPs (b) NTs (c) NSs

Figura 4.1: Muestras de imágenes de la base de datos.

el sistema operativo Windows 11 pro (64 bits). Por parte, en la máquina virtual en la nube, se ejecutaron en un procesador Intel Xeon Scalable de 2ª generación de 3.00 GHz y 24 núcleos, 768 GB de RAM, Aceleradores Gaudi y funcionando bajo un entorno de trabajo Ubuntu.

4.3 Metodología

Como se mencionó anteriormente en esta investigación se siguió la metodología para DL descrita en la sección 2.4.14. A continuación, se describen las actividades realizadas en cada una de sus etapas.

4.3.1 Adquisición de imágenes

Se creó un conjunto de imágenes en tres categorías para evaluar las distintas etapas de la metodología. Para más detalles, véase sección 4.1. Para entrenar con éxito un modelo de DL, se necesitan cientos o incluso miles de imágenes.

4.3.2 Procesamiento de imagen

Para cumplir con los requisitos específicos de las CNN, todas las imágenes de entrada debían tener las mismas características. Sin embargo, las imágenes de la base de datos presentaban resoluciones, formatos y criterios de etiquetado diferentes, lo que hizo necesario realizar diversas operaciones de preprocesamiento. Estas incluyeron el redimensionamiento, la conversión de formatos de archivo, las transformaciones de color y las transformaciones geométricas, como la rotación de las imágenes.

Las imágenes fueron redimensionadas mediante un método de interpolación bicúbica, de manera que todas tuvieran una resolución estándar de 224 x 224 píxeles. Estas imágenes se encontraban en dos formatos principales, con extensiones jpg y tiff. Se optó por estandarizar todas las imágenes al formato jpg. Posteriormente, se aplicó una transformación de color para convertirlas a escala de grises, utilizando la siguiente fórmula:

$$Grises = 0.2989 * R + 0.5870 * G + 0.1140 * B \tag{4.1}$$

donde R corresponde al canal de color rojo, G al canal de color verde, B al canal de color azul, y Grises corresponde a la imagen en escala de grises. En cuanto a las transformaciones geométricas, se utilizaron técnicas como la rotación aleatoria, con un rango de rotación entre 0° y 360°, y el escalado aleatorio, con un factor de escala entre 0.5 y 1.5, con el fin de aumentar la variabilidad en los datos y la robustez del modelo. Es importante tener un buen procesamiento de los datos, para que los parámetros queden lo más óptimo posibles.

4.3.3 Modelado

Tras el procesamiento de las imágenes, el siguiente paso de la metodología es el modelado. En el campo del DL, este paso implica identificar un modelo que pueda clasificar o etiquetar con precisión las imágenes de nuestra base de datos. Para lograrlo, la base de datos se dividió en dos subconjuntos utilizando una partición 70-30. El 70% de las imágenes se usaron para el entrenamiento y el 30% restante para la validación, como se muestra en la tabla 4.1.

Clases	***Imágenes de entrenamiento***	***Imágenes de validación***
NPs	56	24
NTs	56	24
NSs	56	24
Total	168	72

Tabla 4.1: Distribución de la base de datos.

Se realizó una comparación de cuatro modelos de red neuronal convolucional con el objetivo de determinar el mejor de estos. Los cuales fueron una red personalizada (CUSTOM) y tres modelos preexistentes (VGG16, ResNet50 y AlexNet) mencionados en la sección 2.4.3. Los modelos VGG16, ResNet50, AlexNet se seleccionaron por su amplio uso en el campo del DL, destacando eficientemente en la clasificación de imágenes [86].

Nuestro modelo personalizado, recibe imágenes en escala de grises de tres clases en un tamaño de 224 x 224 píxeles, consta de cinco capas con una estructura básica, está compuesto por capas interconectadas, que se muestran en la Fig. 4.2.

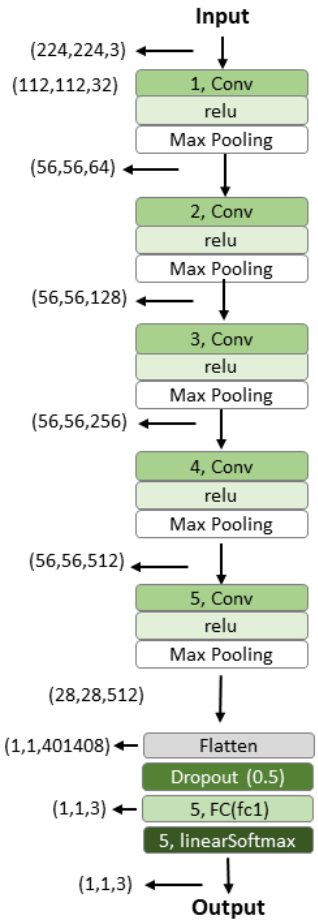


Figura 4.2: Arquitectura de red neuronal CUSTOM.

Para mejorar el rendimiento de dichos modelos, se realizó una búsqueda de cuadrilla para encontrar las mejores combinaciones de los parámetros para cada uno de los modelos, en la que se modificaron la tasa de aprendizaje, los tamaños de los minilotes, las épocas y los optimizadores. En el caso del número de capas, este solo fue modificado para la red CUSTOM. Los valores utilizados para estos parámetros se presentan en la tabla 4.2.

Parámetros	***Valores***
Tasa de aprendizaje	0.01, 0.001, 0.0001
Minilotes	1, 7, 14
Épocas	5, 10, 25
Optimizadores	SGDM, Adam, RMSprop
Capas	3, 4, 5

Tabla 4.2: La configuración de los parámetros se realizó basándose en el enfoque presentado en la sección de hiperparámetros del estado del arte. Estos parámetros fueron cuidadosamente seleccionados teniendo en cuenta las mejores prácticas y metodologías descritas en la literatura existente, con el fin de optimizar el rendimiento del modelo en tareas de clasificación de imágenes.

4.3.4 Evaluación

Una vez identificados los modelos óptimos tanto para el modelo CUSTOM como para los modelos preexistentes, fue crucial validarlos. Para ello, se realizó una validación cruzada aleatoria que constó de diez iteraciones.

Durante este proceso, los datos se dividieron en diez conjuntos diferentes y los modelos se validaron utilizando cada conjunto, obteniendo diez exactitudes para cada modelo, sus matrices de confusión promedio y, a partir de estas, las métricas de exactitud promedio, precisión, sensibilidad y puntaje-F1, con el fin de obtener una medida más fiable.

5. Resultados

Este capítulo presenta los resultados experimentales obtenidos con la metodología descrita en la sección 4.3, así como su discusión. Específicamente, se incluyen los resultados obtenidos mediante la búsqueda de cuadrilla, los resultados generados durante el entrenamiento de los modelos, las gráficas de exactitudes promedio de las arquitecturas, las matrices de confusión y, por último, sus métricas de desempeño.

5.1 Resultados y discusión

Los resultados obtenidos en la búsqueda de cuadrilla, para obtener los mejores parámetros de los modelos se muestran en la tabla 5.1. De acuerdo con los datos de la tabla, la tasa de aprendizaje más empleada por los modelos CUSTOM, VGG16 y AlexNet fue la de 0.0001, comparada con la de ResNet50 que fue de 0.001. Como sabemos, mientras más pequeña sea la tasa de aprendizaje, los modelos aprenden más lento, lo cual puede prevenir el sobreajuste pero también puede representar tiempos de entrenamiento muy prolongados [87].

Por otra parte, el número de minilotes más usado por los modelos VGG16, AlexNet y ResNet50 fue el de 14, en comparación con CUSTOM que usó 7, como se sabe, este está relacionado con la actualización de los pesos y con el número de imágenes necesarias para ello. Los tamaños de lote más pequeños pueden producir tiempos de entrenamiento más largos, generan ruido que ayuda a evitar el sobreajuste, pero también provocan inestabilidad y afectan el equilibrio entre generalización y precisión. Por el contrario, los tamaños de lote más grandes ofrecen una estimación más suave del gradiente, lo que facilita la convergencia hacia una mejor solución. No obstante, también pueden aumentar el riesgo de sobreajuste. Esto puede implicar que los modelos necesiten un mayor número de lotes grandes de imágenes de entrenamiento para actualizar los pesos correctamente, debido a la insuficiencia de imágenes representativas para cada clase [88].

También podemos observar que el número de capas es completamente distinto en cada uno de los modelos, con esto queremos hacer hincapié que para VGG16, AlexNet y ResNet50 el tamaño de capas es predefinido y este no fue modificado por nuestra parte, pero para el modelo CUSTOM, el número de capas seleccionado fue de 5, esto puede deberse a que el número de capas en una red neuronal se decide en función del problema que se está tratando de resolver y de la cantidad de datos disponibles, a medida que aumenta la complejidad del problema y la cantidad de datos, también aumenta el número de capas necesarias.

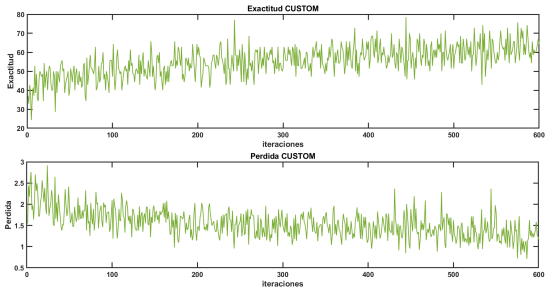
Por último, en cuanto a los optimizadores, el que no fue seleccionado por los modelos fue RMSprop. Esto puede deberse a que dicho optimizador causa un ajuste innecesario de los parámetros en conjuntos de datos pequeños, lo que aumenta el riesgo de sobreajuste [89].

Parámetros	***CUSTOM***	***VGG16***	***AlexNet***	***ResNet50***
Tasa de aprendizaje	0.0001	0.0001	0.0001	0.001
Minilotes	7	14	14	14
Épocas	25	5	10	25
Capas	5	16	8	50
Optimizadores	Adam	SGDM	Adam	SGDM

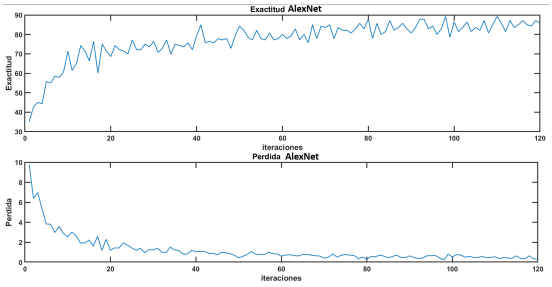
Tabla 5.1: Parámetros de los mejores modelos obtenidos en la búsqueda de cuadrilla.

Durante la validación cruzada aleatoria de diez iteraciones, se generaron datos de exactitud y pérdida, tanto para la etapa de entrenamiento como para la de validación. En esta última etapa, los datos de exactitud y pérdida contenían valores indefinidos (NaN, derivado del inglés, Not a Number), lo cual se debe a un descontrol en la actualización de los pesos (explosión de gradientes) en arquitecturas que tienen un gran número de capas. Los modelos preentrenados aprenden patrones generales de un conjunto de imágenes y al transferir su aprendizaje a un conjunto nuevo con clases diferentes, los valores de los pesos pueden no ser adecuados y generar inestabilidad. Esto impidió la elaboración de los gráficos de validación. Por consiguiente, nos centramos en la discusión de los gráficos de entrenamiento. En la Fig 5.1, se muestran las gráficas promedio (de las diez iteraciones) de exactitud y pérdida de la etapa de entrenamiento de los modelos CUSTOM, VGG16, AlexNet y ResNet50. Respecto a las gráficas de exactitud, los modelos que obtuvieron curvas más suaves (que no presentan saltos grandes y repentinos) fueron VGG16 y AlexNet comparados con el modelo CUSTOM y ResNet50. Esto significa que el desempeño de los modelos (VGG16, AlexNet) cambió más gradualmente y consistentemente conforme avanza la etapa

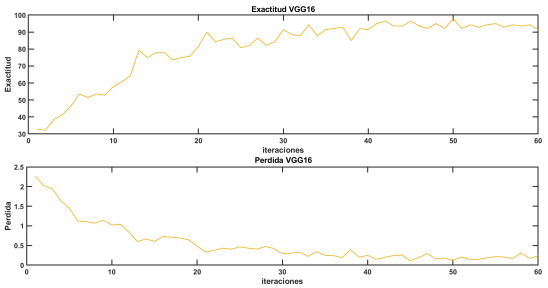
entrenamiento. Otra curva de entrenamiento interesante es la del modelo ResNet50, ya que esta alcanzó las tasas de exactitud más grandes, lo cual se busca en un modelo de aprendizaje profundo. Por otra parte, respecto a las gráficas de pérdida, los modelos AlexNet,VGG16 y ResNet50 tuvieron un comportamiento esperado en comparación con CUSTOM, mostrándonos en sus gráficas que el error tendió a disminuir a medida que progresaba el entrenamiento. Estos modelos tendieron a alcanzar sus puntos de convergencia en distintas iteraciones, lo cual significa que llegaron a un punto en el que, por más que se entrenaran, ya no había cambios. Los modelos que tienden a alcanzar un menor error son VGG16 y ResNet50 contra los modelos CUSTOM y AlexNet. Una recomendación al comparar modelos de aprendizaje profundo es analizar sus curvas de entrenamiento de exactitud y de pérdida, ya que estas proporcionan una representación visual del comportamiento de los modelos.



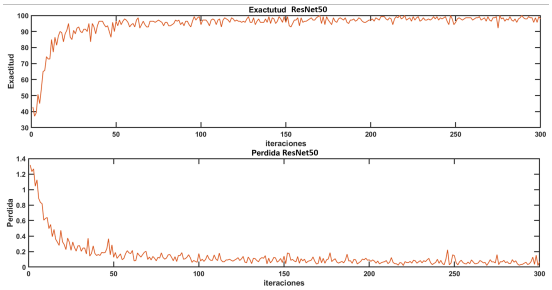
(a) CUSTOM



(b) AlexNet



(c) VGG16



(d) ResNet50

Figura 5.1: Gráficas promedio del progreso de entrenamiento.

En el gráfico de la Fig. 5.2, se muestra el desempeño (exactitud) logrado por los modelos en la validación cruzada aleatoria de diez iteraciones para la etapa de prueba. Es evidente que de los cuatro modelos propuestos, CUSTOM y AlexNet fueron los peores, con exactitudes que oscilan entre 58% y 77% (EP de 63.88%), y entre 65% y 87% (EP de 82.08%), respectivamente. Como puede verse, sus gráficas

de desempeño son variables, en otras palabras, tuvieron un comportamiento que presentó cambios bruscos en términos de exactitud. Recordemos que en los gráficos de desempeño de un buen modelo, se busca tener un comportamiento estable entre iteraciones.

Por otra parte, los mejores modelos fueron VGG16 y ResNet50, los cuales alcanzaron exactitudes que variaron entre 91% y 98% (EP de 95.41%), y entre 91% y 97% (EP de 95.41%) respectivamente. Puede observarse que el comportamiento de sus gráficas es más estable comparado con los modelos CUSTOM y AlexNet, Al ser más estable, casi no presenta cambios de exactitud entre iteraciones. Hasta el momento, los modelos que presentaron los mejores resultados son VGG16 y ResNet50, a continuación se revisan a detalle otras métricas de desempeño con el fin de elegir el mejor modelo de estos.

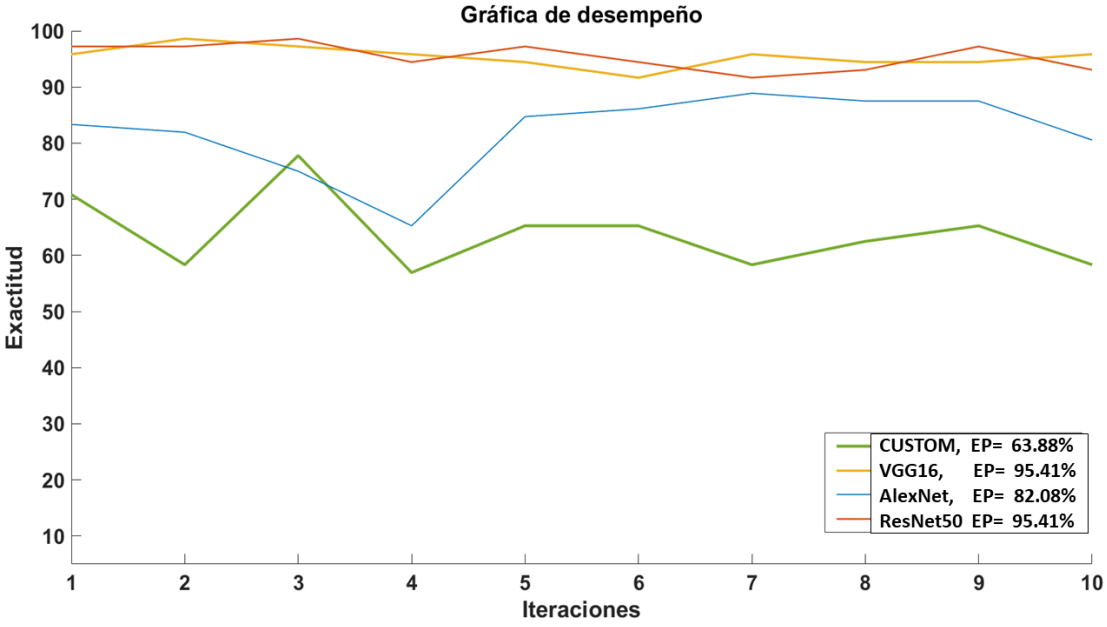
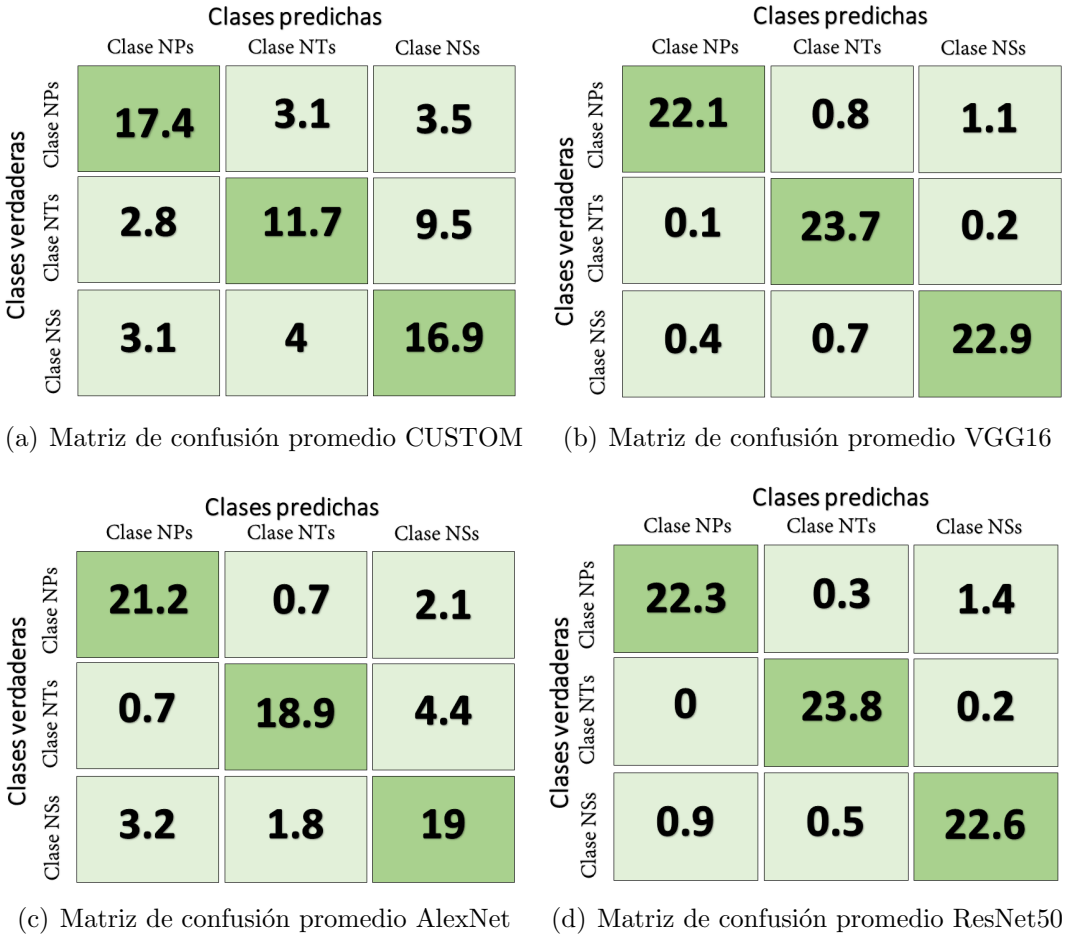


Figura 5.2: Desempeño (exactitud) de los modelos en la validación cruzada aleatoria de diez iteraciones durante la etapa de prueba, EP=exactitud promedio.

A continuación, la Fig. 5.3 muestra las matrices de confusión promedio de nuestros cuatro modelos. Estos valores se calcularon a partir de la validación cruzada aleatoria de 10 iteraciones. Sus renglones representan las clases verdaderas y las columnas las clases predichas. Los valores mostrados en la diagonal principal de las matrices de confusión representan los valores de las clases que han sido predichas de manera correcta.

Para mejorar la evaluación de los modelos, hay que tener en cuenta la incorporación de otras métricas de desempeño como las de la tabla 5.2, las



(a) Matriz de confusión promedio CUSTOM

(b) Matriz de confusión promedio VGG16

(c) Matriz de confusión promedio AlexNet

(d) Matriz de confusión promedio ResNet50

Figura 5.3: Matrices de confusión promedio.

cuales se obtuvieron a partir de las matrices de confusión. En el caso de la precisión promedio, nuevamente los peores modelos fueron CUSTOM y AlexNet, con un 64.47% y un 82.43%, respectivamente. Recordemos que la precisión indica el porcentaje de imágenes de un nanomaterial correctamente identificado con respecto a todas las imágenes clasificadas como tal nanomaterial. Por otra parte, los modelos con mejores valores de precisión fueron ResNet50 con un 95.41% y VGG16 con un 95.48%, con una diferencia no significativa.

En el caso de la sensibilidad, los peores modelos también fueron CUSTOM con un 63.88% y AlexNet con un 82.08%. Recordemos que la sensibilidad es la métrica utilizada para evaluar la capacidad de un modelo para identificar correctamente las imágenes de una clase positiva, respecto al total de imágenes que realmente pertenecen a esa clase. Los mejores valores de sensibilidad promedio

fueron obtenidos por VGG16 y ResNet50 con un 95.41% cada uno.

Además de estas métricas, existe una medida llamada puntaje-F1, que es la combinación de precisión y sensibilidad en un único valor. Esta es de utilidad cuando buscamos encontrar un equilibrio entre precisión y sensibilidad. Para esta métrica, los modelos con valores más bajos fueron CUSTOM y AlexNet con un 63.65% y un 82.12%, respectivamente. Por su parte, los modelos con mejores valores de puntaje-F1 fueron VGG16 y ResNet50 con un 95.40% cada uno. Las métricas anteriores nos dan información sobre el desempeño de los modelos, evaluando su capacidad para clasificar correctamente cada clase. Son indicadores clave, por lo que debemos prestarles igual atención.

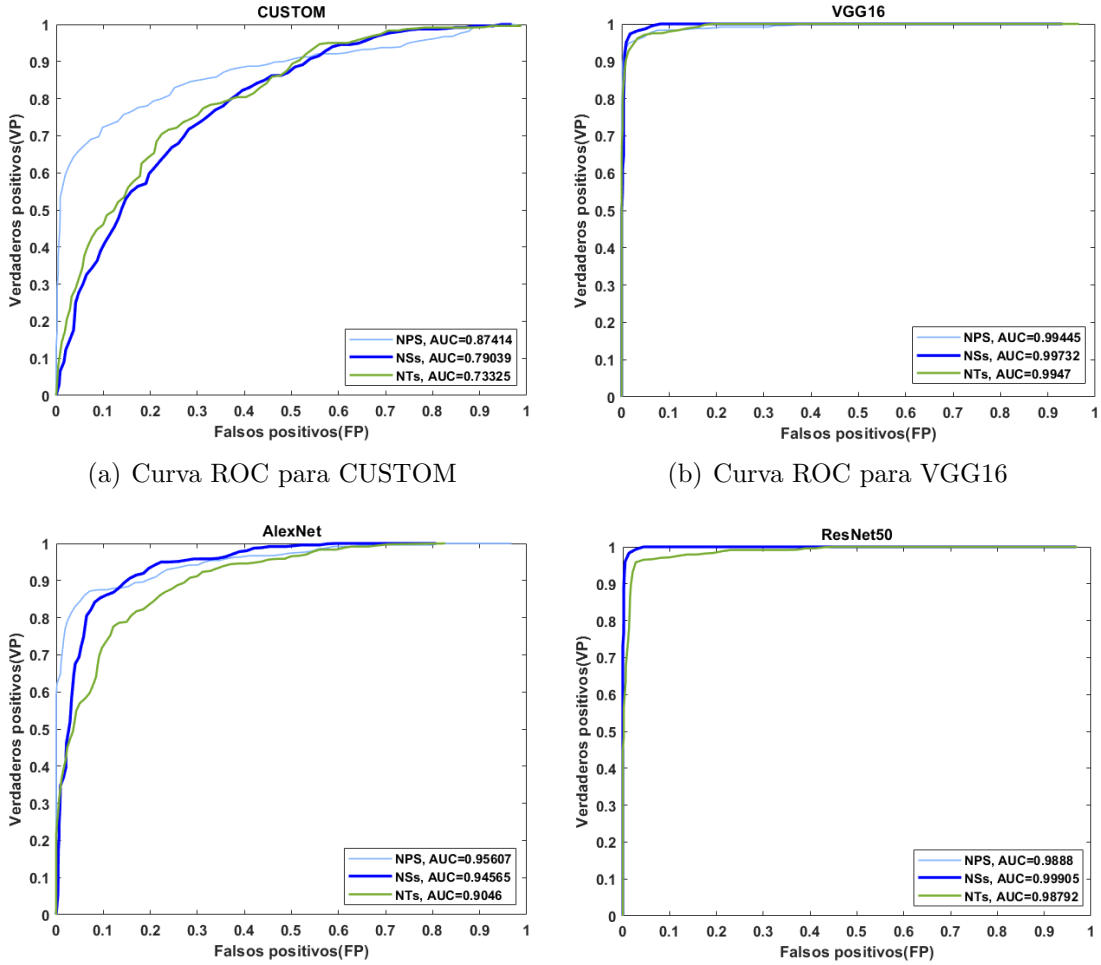
Métricas	***Clases***			***Promedio***
	NPs	***NTs***	***NSs***	
CUSTOM				
Precisión	0.74678	0.62234	0.56522	0.64478
Sensibilidad	0.725	0.4875	0.70417	0.63889
Puntaje-F1	0.73573	0.54673	0.62709	0.63652
AlexNet				
Precisión	0.84862	0.88318	0.7451	0.8243
Sensibilidad	0.88333	0.7875	0.79167	0.82083
Puntaje-F1	0.86354	0.76768	0.76768	0.82127
VGG16				
Precisión	0.97788	0.94048	0.94628	0.95488
Sensibilidad	0.92083	0.9875	0.95417	0.95417
Puntaje-F1	0.9485	0.96341	0.95021	0.95404
ResNet50				
Precisión	0.96121	0.96748	0.93388	0.95419
Sensibilidad	0.92917	0.99167	0.94167	0.95417
Puntaje-F1	0.94492	0.97942	0.93776	0.95403

Tabla 5.2: Métricas de desempeño obtenidas a partir de las matrices de confusión.

En la Fig. 5.4 se muestran las curvas ROC de la etapa de prueba para los modelos CUSTOM, AlexNet, VGG16 y ResNet50; estas nos permiten ver y comparar fácilmente su desempeño en cada una de las clases mediante el AUC, ubicado en la parte inferior derecha de cada gráfico. Recordemos que las curvas ROC muestran el balance existente entre los VP (sensibilidad) y los FP (1 – especificidad). Cuanto más cercana esté la curva de la esquina superior izquierda del gráfico, es decir, más próxima a 1, mejor es el desempeño del modelo para clasificar o diferenciar una clase. Mientras más alejada esté, menor es el desempeño.

Como era de esperarse, los modelos que tuvieron el peor desempeño en función de sus curvas ROC fueron CUSTOM y AlexNet. Por otro lado, los mejores modelos fueron VGG16 y ResNet50. Con respecto a la clase NPs, el modelo que logró clasificarla mejor fue VGG16, con un AUC de 0.994, en comparación con un 0.988 de ResNet50. En el caso de la clase NSs, el modelo que consiguió clasificarla mejor con un AUC de 0.999 fue ResNet50, en relación con VGG16 que obtuvo un AUC de 0.997. Y finalmente, para la clase NTs, el modelo que logró clasificarla mejor fue VGG16 con un AUC de 0.994, respecto a ResNet50 que obtuvo un AUC de 0.987. En suma, el modelo VGG16 fue capaz de clasificar mejor dos de las tres clases.

De acuerdo con la información revisada, no existe una diferencia significativa en los resultados obtenidos por los modelos VGG16 y ResNet50. Si se desea realizar más experimentos para mejorar las tasas de reconocimiento en imágenes de nanomateriales, se recomienda el modelo VGG16 sobre ResNet50, dado que este modelo tiene un número de parámetros muy inferior y permite realizar entrenamientos en mucho menos tiempo. Con base en los resultados de esta investigación en clasificación de imágenes de nanomateriales, se recomienda una arquitectura VGG16 con tasa de aprendizaje de 0.0001, un tamaño de minilote de 14, un número de épocas igual a 5 y un optimizador sgdm, ya que esta combinación de parámetros logró una tasa de reconocimiento del 95.41%.



(a) Curva ROC para CUSTOM

(b) Curva ROC para VGG16

(c) Curva ROC para AlexNet

(d) Curva ROC para ResNet50

Figura 5.4: Curvas ROC de los modelos.

En la tabla 5.3 se muestran algunas imágenes del conjunto de datos utilizado por los modelos. A la izquierda se presentan las imágenes con sus etiquetas correctas, mientras que a la derecha se muestran las etiquetas predichas, acompañadas de algunas imágenes de referencia para ilustrar las razones por las cuales los modelos están confundiendo las clases.

Las imágenes de los incisos (a) y (b) revelan que, cuando la magnificación entre NPs y NTs es parecida, existe una similitud morfológica entre las muestras, que los modelos de aprendizaje profundo no son capaces de discriminar. En otras palabras, si se adquiere una imagen muy alejada de la muestra, donde las NPs no se logran apreciar en su forma completa, se genera una confusión y por lo tanto una mala predicción de los modelos. En el caso de las imágenes (c) y (d), ocurre algo similar con los NSs, estos tienden a ser confundidos con las NPs por nuestras arquitecturas. Finalmente, en las imágenes (e) y (f), se puede apreciar que los NSs tienen una similitud morfológica con los NTs verticalmente alineados, debido a la perspectiva con la que fueron adquiridas las imágenes, dicha similitud genera confusión en los modelos a la hora de clasificar estos nanomateriales.

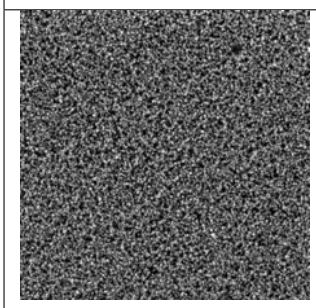
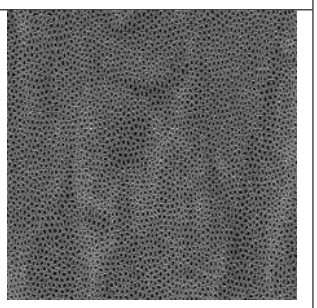
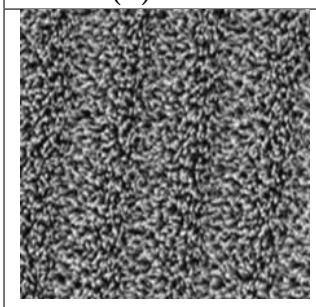
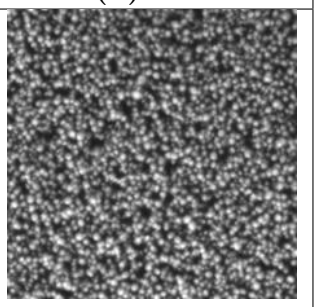
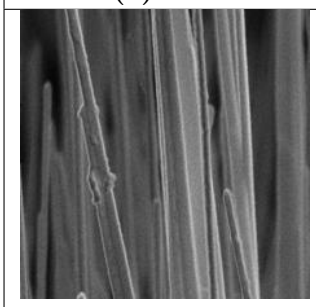
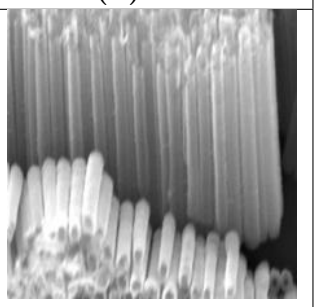
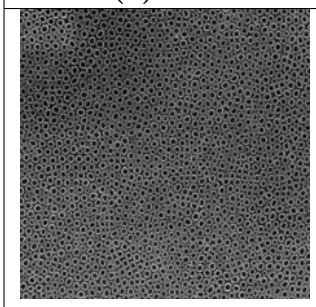
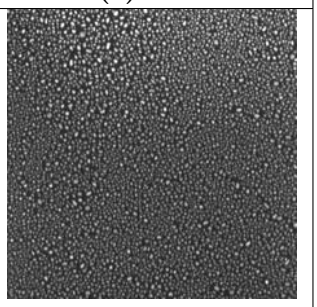
Verdad	Predicción
	
(a) NPs	**(b)** NTs
	
(c) NSs	**(d)** NPs
	
(e) NSs	**(f)** NTs
	
(g) NTs	**(h)** NPs

Tabla 5.3: Muestra de imágenes mal clasificadas por los modelos (**verdad frente a predicción**).

6. Conclusiones

La falta de un estándar de etiquetado de imágenes (SEM) de nanomateriales dificulta la gestión de estas, por ejemplo, en la construcción de bases de datos para experimentación. En este contexto, el DL surge como una solución prometedora, al permitir el desarrollo de modelos capaces de clasificar eficientemente imágenes generadas mediante microscopios electrónicos. Con base en los resultados logrados en esta investigación sobre clasificación de imágenes SEM a partir de técnicas de aprendizaje profundo, se concluye lo siguiente.

Se demostró que el DL es una alternativa viable en el desarrollo de herramientas y sistemas complementarios para la clasificación de imágenes de diversos nanomateriales, lo cual apoyaría a los métodos de organización tradicionales. Estas herramientas permitirían una mejor gestión de los datos, facilitando su explotación tanto a corto como a largo plazo. Además, contribuirían en la creación de un estándar para su etiquetado.

Se realizó una comparación de cuatro modelos de aprendizaje profundo: uno personalizado (CUSTOM) y tres modelos preexistentes (AlexNet, VGG16 y ResNet50). Los resultados obtenidos mostraron que tanto CUSTOM como AlexNet fueron los peores modelos, ya que presentaron exactitudes y puntajes-F1 promedio menores. Se considera que esto es debido a una cantidad insuficiente de capas, que no les permite abordar adecuadamente la complejidad del problema. En cambio, VGG16 y ResNet50 fueron los mejores modelos, con exactitudes promedio del 95.41% y puntajes-F1 promedio de 0.9540. Cabe señalar que no hubo una diferencia significativa en los resultados obtenidos por estos modelos.

Se recomienda utilizar la arquitectura VGG16 con optimizador SGDM para ejercicios de clasificación de imágenes de nanomateriales, con una tasa de aprendizaje igual a 0.0001, un tamaño de minilote igual a 14 y un número de épocas igual a 5, porque con estos parámetros se obtuvieron los mejores resultados en la experimentación. Una de las principales razones para recomendar esta arquitectura sobre Resnet50 es su número de capas, lo que se traduce en tiempos de entrenamiento más rápidos y la posibilidad de seguir mejorando el modelo.

Este balance entre la profundidad del modelo y la complejidad computacional optimiza el rendimiento, reduciendo la carga en el proceso de entrenamiento sin comprometer la capacidad de aprendizaje del modelo.

En la realización de este trabajo de investigación se presentaron ciertas limitaciones. Cuando se trabaja con arquitecturas de DL, se suele requerir de una gran cantidad de datos para ejercicios de clasificación. Por lo tanto, una de las desventajas de la investigación se presenta con la falta de una base de datos adecuada para la experimentación, esto pudo impedir la obtención de mejores resultados por parte de los modelos. Otra limitante es que por un largo período de tiempo no se dispuso de un equipo de cómputo con mejores recursos (como, el cómputo en la nube), lo cual no permitió una experimentación con un ajuste más fino de los parámetros de los modelos.

Para mejorar la clasificación de imágenes de nanomateriales, se pueden considerar las siguientes estrategias como trabajo futuro. Experimentar con formatos de imagen sin compresión como PGM y GIF, porque mantienen la calidad de los datos y permitirían un procesamiento más rápido de estos. Realizar una comparación de los modelos con imágenes de buena y mala calidad para medir su robustez e incluir arquitecturas modernas como SRCNN, EfficientNet y Vision Transformers. Implementar los mejores modelos en aplicaciones para el uso de laboratorios de nanomateriales y validar si mejoran la gestión de grandes volúmenes de datos. Ampliar la base de datos para experimentación y aplicar nuevas técnicas de optimización y búsqueda de parámetros para mejorar la precisión y eficiencia de los modelos.

7. Anexos A

Implementación de la metodología

A continuación, se presenta una implementación de los algoritmos de la metodología para DL propuesta en este trabajo utilizando el software descrito en la sección 4.2. Estos códigos están cubiertos bajo la Licencia Pública General de GNU (GPL) versión 3 y se pueden consultar en el siguiente repositorio público disponible en github: `https://github.com/LuisQUINTERO0/tecnicas-de-Aprendizaje-profundo-para-clasificacion-Nanomateriales-2024` .

7.1 Procesamiento de imagen

Este código se utiliza en la etapa número tres, correspondiente al procesamiento de imágenes, descrita en la Sección 4.3.2.

```
1
2 *Specific route*
3 %Specifies the main folder where the subfolders with images are
      located.
4 main_folder = 'C:\Users\Desktop\NMs';
5
6 % Getting all subfolders inside the main folder.
7 subfolders = dir(main_folder);
8 subfolders = subfolders([subfolders.isdir]);  % Filter only
     folders.
9
10 % Counter to number the images.
11 counter_images = 1;
12
13 % Scrolls each subfolder
14 for i = 1:numel(subfolders)
```

```
15      name_subfolder = subfolders(i).name;
16      if ~strcmp(name_subfolder, '.') && ~strcmp(name_subfolder,
            '..')
17
18  % Reads all images inside the current subfolder.
19          current_folder = fullfile(main_folder, name_subfolder);
20          image_files = dir(fullfile(current_folder, '*.jpg'));
                % Change format according to your images.
21
22  % Creates a folder to store the resized images.
23          destination_folder = fullfile(main_folder,'
                resized_images',name_subfolder);
24          if ~exist(destination_folder, 'dir')
25              mkdir(destination_folder);
26          end
27
28  % Iterate over each image file.
29          for j = 1:numel(image_files)
30              image_name = image_files(j).name;
31              image_path = fullfile(current_folder, image_name);
32
33  *Pre-preparedness on images*
34
35  % Read the image.
36              image = imread(image_path);
37  % Define the dimensions of the images.
38              new_height = 224;
39              new_width = 224;
40              resized_image = imresize(image, [new_height,
                    new_width]);
41              counter_images = counter_images + 1;
42
43  *Saving of images*
44
45              % Saves images individually according to their
                    folder name.
46              [~, base_name, ~] = fileparts(image_name);  %
                    Extracts the base name without extension.
47              file_name = [subfolder_name '_' base_name '.jpg'];
48              % File name with label, extension can be changed if
                     desired.
49
50              % Save the resized image inside the destination
                    folder.
51              saved_route = fullfile(destination_folder,
                    file_name);
52              imwrite(resized_image, saved_route);
53          end
54      end
```

```
55 end
56 disp('Finshed.');
```

7.2 Búsqueda de cuadrilla

Este código se utiliza en la etapa número cuatro, correspondiente al modelado, descrita en la Sección 4.3.3.

```
1  tic
2  fprintf("\nStarting....\n")
3  %% Separation of data.
4  imds = imageDatastore('C:\Users\Desktop\New folder\
       GridSearchFinal',...
5      'FileExtensions',{'.jpg'},...
6      'IncludeSubfolders',true,'LabelSource','foldernames')
7  [imdsTrain,imdsValidation] = splitEachLabel(imds,0.7,'
       randomized');
8
9  %% Data augmentation.
10 imageAugmenter = imageDataAugmenter( ...
11     'RandRotation',[0,360], ...
12     'RandScale',[0.5,1.5])
13 inputSize = [224 224 3];
14 augimdsTrain = augmentedImageDatastore(inputSize(1:3),imdsTrain
       ,'DataAugmentation',imageAugmenter);
15 augimdsValidation = augmentedImageDatastore(inputSize(1:3),
       imdsValidation);
16
17 N=[0.01,0.001,0.0001];
18 minibatch=[1,7,14];
19 Epoch=[5,10,25];
20 optmz={'sgdm','adam','rmsprop'};
21 layerss=[3,4,5];
22 accuracyVGG16=     zeros(3,3,3,3);% 81
23 accuracyResNet50= zeros(3,3,3,3); %243
24 accuracyCUSTOM=   zeros(3,3,3,3,3);
25 accuracyAlexNet=  zeros(3,3,3,3);
26 for i=1:length(N)
27     for j=1:length(minibatch)
28         for  k=1:length(Epoch)
29             for  l=1:length(optmz)
30                 try
31                 [accuracyVGG16(i,j,k,l),~,~,~,~]= VGG16(
                       imdsTrain,imdsValidation,augimdsValidation,
                       augimdsTrain,N(i),minibatch(j),Epoch(k),
                       optmz(l))
```

```
32                  catch  ME
33                      disp(ME.message)
34                      accuracyVGG16(i,j,k,l)=0;
35                  end
36                  try
37                      [accuracyResNet50(i,j,k,l),~,~,~,~]=
                            ResNet50(imdsTrain,imdsValidation,
                            augimdsValidation,augimdsTrain,N(i),
                            minibatch(j),Epoch(k),optmz(l))
38                  catch ME
39                      disp(ME.message)
40                      accuracyResNet50(i,j,k,l)=0;
41                  end
42                  try
43                      [accuracyAlexNet(i,j,k,l),~,~,~,~]= AlexNet
                            (imdsTrain,imdsValidation,
                            augimdsValidation,augimdsTrain,N(i),
                            minibatch(j),Epoch(k),optmz(l))
44                  catch ME
45                      disp(ME.message)
46                      accuracyAlexNet(i,j,k,l)=0;
47                  end
48
49                  for m=1:length(layerss)
50                      try
51
52                          [accuracyCUSTOM(i,j,k,l,m),~,~,~,~]=
                                CUSTOM(imdsTrain,imdsValidation,
                                augimdsValidation,augimdsTrain,
                                inputSize,N(i),minibatch(j),Epoch(k)
                                ,optmz(l),m);
53                      catch
54                          accuracyCUSTOM(i,j,k,l,m)=0;
55                      end
56                  end
57              end
58          end
59      end
60  end
61  %% Obtaining the best parameters for the CUSTOM model.
62  [M,I]=max(accuracyCUSTOM,[],"all","linear");
63  [i,j,k,l,m]=ind2sub(size(accuracyCUSTOM),I);
64  Ncustom=N(i(1));
65  minibatchcustom=minibatch(j(1));
66  Epochcustom=Epoch(k(1));
67  optmzcustom=optmz(l(1));
68  layersscustom=layerss(m(1));
69
70  %% Obtaining the best parameters for the VGG16 model.
```

```
[M1,I]=max(accuracyVGG16,[],"all","linear");
[i,j,k,l]=ind2sub(size(accuracyVGG16),I);
NVGG16=N(i(1));
minibatchVGG16=minibatch(j(1));
EpochVGG16=Epoch(k(1));
optmzVGG16=optmz(l(1));

%% Obtaining the best parameters for the ResNet50 model.
[R1,H]=max(accuracyResNet50,[],"all","linear");
[i,j,k,l]=ind2sub(size(accuracyResNet50),H);
NResNet50=N(i(1));
minibatchResNet50=minibatch(j(1));
EpochResNet50=Epoch(k(1));
optmzResNet50=optmz(l(1));

%% Obtaining the best parameters for the AlexNet model.
[a1,n]=max(accuracyAlexNet,[],"all","linear");
[i,j,k,l]=ind2sub(size(accuracyAlexNet),n);
NAlexNet=N(i(1));
minibatchAlexNet=minibatch(j(1));
EpochAlexNet=Epoch(k(1));
optmzAlexNet=optmz(l(1));

fprintf("Finished...\n")
tEnd=toc;
h=tEnd/60;
fprintf(' %d hours and %d minutes and %f seconds\n',floor(h/
    60), floor(tEnd/60), rem(tEnd,60));
```

7.3 Validación cruzada aleatoria

Este código se utiliza en la etapa número cinco, correspondiente a la evaluación, descrita en la Sección 4.3.4.

```
%tic
fprintf("\nStarting....\n")
accuracyCUSTOM  =zeros(1,10);
accuracyVGG16   =zeros(1,10);
accuracyResNet50=zeros(1,10);
accuracyAlexNet =zeros(1,10);

cmCUSTOM = zeros(3,3,10);
cmVGG16 = zeros(3,3,10);
cmResNet50 = zeros(3,3,10);
cmAlexNet = zeros(3,3,10);

```

```
infoCUSTOM = cell(1,10);
infoVGG16 = cell(1,10);
infoResNet50 = cell(1,10);
infoAlexNet = cell(1,10);

rocObjCUSTOM = cell(1,10);
rocObjVGG16 = cell(1,10);
rocObjResNet50 = cell(1,10);
rocObjAlexNet = cell(1,10);

%% Separation of data.
imds = imageDatastore('C:\Users\Desktop\New folder\
    CrossValidation',...
    'FileExtensions',{'.jpg'},...
    'IncludeSubfolders',true,'LabelSource','foldernames')
%% Data augmentation.
imageAugmenter = imageDataAugmenter( ...
    'RandRotation',[0,360], ...
    'RandScale',[0.5,1.5])
inputSize = [224 224 3];

for i=1:10
    [imdsTrain,imdsValidation] = splitEachLabel(imds,0.7,'
        randomized');
    augimdsTrain = augmentedImageDatastore(inputSize(1:3),
        imdsTrain,'DataAugmentation',imageAugmenter);
    augimdsValidation = augmentedImageDatastore(inputSize(1:3),
        imdsValidation);

    %%1. Random cross validation for CUSTOM.
    try
    [accuracyCUSTOM(i),YValidationCUSTOM,YPredCUSTOM,info,
        rocObj]=CUSTOM(imdsTrain,imdsValidation,
        augimdsValidation,augimdsTrain,inputSize,0.0001,7,25,"
        adam",3);
    cmCUSTOM(:,:,i) = confusionmat(YValidationCUSTOM,
        YPredCUSTOM);
    infoCUSTOM{i} = info;
    rocObjCUSTOM{i} = rocObj;
    catch
        accuracyCUSTOM(i),YValidationCUSTOM,YPredCUSTOM,info,
            rocObj=0
    end

    %%2. Random cross validation for VGG16.
    try
    [accuracyVGG16(i),YValidationVGG16,YPredVGG16,info,rocObj]=
        VGG16(imdsTrain,imdsValidation,augimdsValidation,
        augimdsTrain,0.0001,14,5,"sgdm");
```

```
51      cmVGG16(:,:,i) = confusionmat(YValidationVGG16,YPredVGG16);
52      infoVGG16{i} = info;
53      rocObjVGG16{i} = rocObj;
54      catch
55       accuracyVGG16(i),YValidationVGG16,YPredVGG16,info,rocObj=0
56      end
57
58      %%3. Random cross validation for ResNet50.
59      try
60      [accuracyResNet50(i),YValidationResNet50,YPredResNet50,info
           ,rocObj]= ResNet50(imdsTrain,imdsValidation,
           augimdsValidation,augimdsTrain,0.001,14,25,"sgdm");
61      cmResNet50(:,:,i) = confusionmat(YValidationResNet50,
           YPredResNet50);
62      infoResNet50{i} = info;
63      rocObjResNet50{i} = rocObj;
64      catch
65          accuracyResNet50(i),YValidationResNet50,YPredResNet50,
               info,rocObj=0
66      end
67
68      %%4. Random cross validation for AlexNet.
69      try
70      [accuracyAlexNet(i),YValidationAlexNet,YPredAlexNet,info,
           rocObj]= AlexNet(imdsTrain,imdsValidation,
           augimdsValidation,augimdsTrain,0.0001,14,10,"adam");
71      cmAlexNet(:,:,i) = confusionmat(YValidationAlexNet,
           YPredAlexNet);
72      infoAlexNet{i} = info;
73      rocObjAlexNet{i} = rocObj;
74      catch
75          accuracyAlexNet(i),YValidationAlexNet,YPredAlexNet,info
               ,rocObj=0
76      end
77  end
78  fprintf("Finished...\n")
79  tEnd=toc;
80  h=tEnd/60;
81  fprintf(' %d hours and %d minutes and %f seconds\n',floor(h/
       60), floor(tEnd/60), rem(tEnd,60));
```

Referencias

[1] S. Machado, J. Pacheco, H. Nouws, J. T. Albergaria, and C. Delerue-Matos, "Characterization of green zero-valent iron nanoparticles produced with tree leaf extracts," *Science of the total environment*, vol. 533, pp. 76–81, 2015. 1

[2] J. Amaya and W. Quiroga, "Nanomateriales: una clasificación desde sus dimensiones," *Revista Química e Industria*, pp. 10–12, 2019. 1

[3] M. Gómez-Garzón, "Nanomateriales, nanopartículas y síntesis verde," *Rev. Repert. Med. Cir.*, vol. 27, no. 2, 2018. 1

[4] R. Shoukat and M. I. Khan, "Carbon nanotubes: a review on properties, synthesis methods and applications in micro and nanotechnology," *Microsystem Technologies*, pp. 1–10, 2021. 1

[5] J. Zhang, Y. Coffinier, Z.-Y. Zhao, S. Szunerits, A. Barras, X. Yu, and R. Boukherroub, "Preparation of boron-doped diamond nanospikes on porous ti substrate for high-performance supercapacitors," *Electrochimica Acta*, vol. 354, p. 136649, 2020. 1

[6] J. Huo, Q. Jia, K. Wang, J. Chen, J. Zhang, P. Li, and W. Huang, "Metal-phenolic networks assembled on tio2 nanospikes for antimicrobial peptide deposition and osteoconductivity enhancement in orthopedic applications," *Langmuir*, vol. 39, no. 3, pp. 1238–1249, 2023. 1

[7] K. Mukaddam, M. Astasov-Frauenhoffer, E. Fasler-Kan, L. Marot, M. Kisiel, R. Steiner, F. Sanchez, E. Meyer, J. Köser, M. M. Bornstein *et al.*, "Novel titanium nanospike structure using low-energy helium ion bombardment for the transgingival part of a dental implant," *Nanomaterials*, vol. 12, no. 7, p. 1065, 2022. 1

[8] R. Tripathi and S. J. Chung, "Biogenic nanomaterials: Synthesis, characterization, growth mechanism, and biomedical applications," *Journal*

of Microbiological Methods, vol. 157, pp. 65–80, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S016770121830383X 1

[9] K. Mukaddam, M. Astasov-Frauenhoffer, E. Fasler-Kan, L. Marot, M. Kisiel, R. Steiner, F. Sanchez, E. Meyer, J. Köser, M. M. Bornstein, and S. Kühl, "Novel titanium nanospike structure using low-energy helium ion bombardment for the transgingival part of a dental implant," *Nanomaterials*, vol. 12, no. 7, 2022. [Online]. Available: https://www.mdpi.com/2079-4991/12/7/1065 1

[10] G. Chen, J. Seo, C. Yang, and P. N. Prasad, "Nanochemistry and nanomaterials for photovoltaics," *Chem. Soc. Rev.*, vol. 42, pp. 8304–8338, 2013. [Online]. Available: http://dx.doi.org/10.1039/C3CS60054H 1

[11] S. Ameen, M. S. Akhtar, and H. S. Shin, "Growth and characterization of nanospikes decorated zno sheets and their solar cell application," *Chemical engineering journal*, vol. 195, pp. 307–313, 2012. 1

[12] M. H. Modarres, R. Aversa, S. Cozzini, R. Ciancio, A. Leto, and G. P. Brandino, "Neural network for nanoscience scanning electron microscope image recognition," *Scientific reports*, vol. 7, no. 1, p. 13282, 2017. 2

[13] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015. 4

[14] F. López de la Rosa, R. Sánchez-Reolid, J. L. Gómez-Sirvent, R. Morales, and A. Fernández-Caballero, "A review on machine and deep learning for semiconductor defect classification in scanning electron microscope images," *Applied Sciences*, vol. 11, no. 20, 2021. [Online]. Available: https://www.mdpi.com/2076-3417/11/20/9508 4

[15] G. Algan and I. Ulusoy, "Image classification with deep learning in the presence of noisy labels: A survey," *Knowledge-Based Systems*, vol. 215, p. 106771, 2021. 4

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778. 4

[17] Á. Camacho and M. Zapata, "Qué es un nanomaterial," *"Momento"*, no. "54E", pp. "57–66", "2017". 6

[18] I. César and C. Mendoza, "Nanomateriales." 6

[19] G. G. Etxebarria, "Seguridad y salud en el trabajo con nanomateriales." *Gestión práctica de riesgos laborales: Integración y desarrollo de la gestión de la prevención*, no. 145, pp. 36–49, 2017. 7

[20] S. Haykin, *Neural networks and learning machines, 3/E.* Pearson Education India, 2009. 11

[21] J. Siverio Rojas, "Deep learning en interfaces conversacionales," 2022. 13

[22] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning.* MIT Press, 2016, book in preparation for MIT Press. [Online]. Available: http://www.deeplearningbook.org 13

[23] J. L. Sarmiento-Ramos, "Aplicaciones de las redes neuronales y el deep learning a la ingeniería biomédica," *Revista UIS Ingenierías*, vol. 19, no. 4, pp. 1–18, 2020. 13

[24] F. M. Tipantocta, "Red neuronal convolucional aplicado a la estimulación de motricidad fina con un videojuego en un paciente con hemiparesia izquierda," *Cumbres*, vol. 6, no. 1, pp. 23–32, 2020. 14

[25] F. Rozo-García, "Revisión de las tecnologías presentes en la industria 4.0," *Revista UIS Ingenierías*, vol. 19, no. 2, pp. 177–191, 2020. 14

[26] D. Quirumbay Yagual, C. Castillo Yagual, and I. Coronel Suárez, "Una revisión del aprendizaje profundo aplicado a la ciberseguridad," *Revista Científica y Tecnológica UPSE*, vol. 9, no. 1, pp. 57–65, jun. 2022. 14

[27] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015. 15

[28] F. Chollet, *Deep learning with Python.* Simon and Schuster, 2021. 16

[29] D. Britz, "Understanding convolutional neural networks for NLP," 2015. 18

[30] J. M. Vega Arias, "Modelo de aprendizaje profundo/red neuronal convolucional (cnn) para clasificación de calidad de ácidos grasos por imágenes de semillas de helianthus annuus," 2019. 19

[31] Y.-L. Boureau, F. Bach, Y. LeCun, and J. Ponce, "Learning mid-level features for recognition," in *2010 IEEE computer society conference on computer vision and pattern recognition.* IEEE, 2010, pp. 2559–2566. 19

[32] S. S. Basha, S. R. Dubey, V. Pulabaigari, and S. Mukherjee, "Impact of fully connected layers on performance of convolutional neural networks for image classification," *Neurocomputing*, vol. 378, pp. 112–119, 2020. 20

[33] X. Ding, C. Xia, X. Zhang, X. Chu, J. Han, and G. Ding, "Repmlp: Re-parameterizing convolutions into fully-connected layers for image recognition," *arXiv preprint arXiv:2105.01883*, "2021". 20

[34] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014. 23

[35] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Clasificación de imagenet con redes neuronales convolucionales profundas," *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097–1105, 2012. 24

[36] T. Lu, B. Han, and F. Yu, "Detection and classification of marine mammal sounds using alexnet with transfer learning," *Ecological Informatics*, vol. 62, p. 101277, 2021. 24

[37] A. S. B. Reddy and D. S. Juliet, "Transfer learning with resnet-50 for malaria cell-image classification," in *2019 International conference on communication and signal processing (ICCSP)*. IEEE, 2019, pp. 0945–0949. 24

[38] H. H. Hoang and H. H. Trinh, "Improvement for convolutional neural networks in image classification using long skip connection," *Applied Sciences*, vol. 11, no. 5, p. 2092, 2021. 25

[39] K. Akshai and J. Anitha, "Plant disease classification using deep learning," in *2021 3rd International Conference on Signal Processing and Communication (ICPSC)*. IEEE, 2021, pp. 407–411. 28

[40] D. Müller, I. Soto-Rey, and F. Kramer, "Towards a guideline for evaluation metrics in medical image segmentation," *BMC Research Notes*, vol. 15, no. 1, p. 210, 2022. 28

[41] D. Chicco and G. Jurman, "The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation," *BMC genomics*, vol. 21, pp. 1–13, 2020. 29

[42] K. Alomar, H. I. Aysel, and X. Cai, "Data augmentation in classification and segmentation: A survey and new strategies," *Journal of Imaging*, vol. 9, no. 2, p. 46, 2023. 31

[43] J. Jang, D. Kyung, S. H. Kim, H. Lee, K. Bae, and E. Choi, "Significantly improving zero-shot x-ray pathology classification via fine-tuning pre-trained image-text encoders," *Scientific Reports*, vol. 14, no. 1, p. 23199, 2024. 31

[44] H. E. Kim, A. Cosa-Linan, N. Santhanam, M. Jannesari, M. E. Maros, and T. Ganslandt, "Transfer learning for medical image classification: a literature review," *BMC medical imaging*, vol. 22, no. 1, p. 69, 2022. 32

[45] D. M. Belete and M. D. Huchaiah, "Grid search in hyperparameter optimization of machine learning models for prediction of hiv/aids test results," *International Journal of Computers and Applications*, vol. 44, no. 9, pp. 875–886, 2022. 32

[46] M. W. Browne, "Cross-validation methods," *Journal of mathematical psychology*, vol. 44, no. 1, pp. 108–132, 2000. 33

[47] D. Berrar *et al.*, "Cross-validation." 2019. 33

[48] J. M. Pérez and P. P. Martin, "La curva roc," *Medicina de Familia. SEMERGEN*, vol. 49, no. 1, p. 101821, 2023. 34

[49] B. Gawin, R. Małkowski, and R. Rink, "Will nilm technology replace multi-meter telemetry systems for monitoring electricity consumption?" *Energies*, vol. 16, no. 5, p. 2275, 2023. 34

[50] Q. Luo, E. A. Holm, and C. Wang, "A transfer learning approach for improved classification of carbon nanomaterials from tem images," *Nanoscale Advances*, vol. 3, no. 1, pp. 206–213, 2021. 38

[51] K. Han, J. He, Y. Wang, Y. Xiong, and C. Zhang, "An image classification approach based on deep learning and transfer learning," in *IOP Conference Series: Materials Science and Engineering*, vol. 768, no. 7. IOP Publishing, 2020, p. 072055. 39

[52] M. Shaha and M. Pawar, "Transfer learning for image classification," in *2018 second international conference on electronics, communication and aerospace technology (ICECA)*. IEEE, 2018, pp. 656–660. 39

[53] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," 2007. 39

[54] J. Li and J. Z. Wang, "Automatic linguistic indexing of pictures by a statistical modeling approach," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 25, no. 9, pp. 1075–1088, 2003. 39

[55] M. X. Bastidas-Rodriguez, L. F. Polanía, A. Gruson, and F. Prieto-Ortiz, "Deep learning for fractographic classification in metallic materials," *Engineering Failure Analysis*, 2020. 39

[56] I. E. Cicero, "Utilización de redes neuronales convoluciones para la detección de tipos de imágenes," 2018. 40

[57] J. M. Ede, "Deep learning in electron microscopy," *Machine Learning: Science and Technology*, vol. 2, no. 1, 2021, cited by: 16; All Open Access, Gold Open Access, Green Open Access. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85101464715&doi=10.1088%2f2632-2153%2fabd614&partnerID=40&md5=0314ae54375d899eecd46cb3ed417ab5 40

[58] Y. Kakishita, A. Ray, H. Hattori, A. Hisada, Y. Ominami, J.-P. Baudoin, J. Y. B. Khalil, and D. Raoult, "Quantitative analysis system for bacterial cells in sem image using deep learning," in *2021 55th Annual Conference on Information Sciences and Systems (CISS)*, March 2021, pp. "01–06". 40

[59] D. Kolenov, D. Davidse, J. Le Cam, and S. Pereira, "Convolutional neural network applied for nanoparticle classification using coherent scatterometry data," *Applied Optics*, vol. 59, no. 27, pp. 8426–8433, 2020. 41

[60] J. Na, G. Kim, S.-H. Kang, S.-J. Kim, and S. Lee, "Deep learning-based discriminative refocusing of scanning electron microscopy images for materials science," *Acta Materialia*, vol. 214, 2021, cited by: 8. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85107088879&doi=10.1016%2fj.actamat.2021.116987&partnerID=40&md5=fb0ef7255613759484ff8970568eb8be 41

[61] C. Shen, C. Wang, M. Huang, N. Xu, S. van der Zwaag, and W. Xu, "A generic high-throughput microstructure classification and quantification method for regular sem images of complex steel microstructures combining ebsd labeling and deep learning," *Journal of Materials Science Technology: an international journal in the field of materials science*, vol. 93, pp. 191–204, 2021, green Open Access added to TU Delft Institutional Repository 'You share, we take care!' – Taverne project https://www.openaccess.nl/en/you-share-we-take-care Otherwise as indicated in the copyright section: the publisher is the copyright holder of this work and the author uses the Dutch legislation to make this work public. 42

[62] V. Biyyapu, Y. V. Reddy, S. K. Reddy, and P. Priyadarshini, "Object detection using opencv," in *AIP Conference Proceedings*, vol. 2742, no. 1. AIP Publishing, 2024. 42

[63] J. L. Vincent, R. Manzorro, S. Mohan, B. Tang, D. Y. Sheth, E. P. Simoncelli, D. S. Matteson, C. Fernandez-Granda, and P. A. Crozier, "Developing and evaluating deep neural network-based denoising for nanoparticle tem images with ultra-low signal-to-noise," *Microscopy and Microanalysis*, vol. 27, no. 6, pp. 1431–1447, 2021. 42

[64] P. Napoletano, F. Piccoli, and R. Schettini, "Anomaly detection in nanofibrous materials by cnn-based self-similarity," *Sensors*, vol. 18, no. 1, p. 209, 2018. 43

[65] C. Ieracitano, F. Pantò, N. Mammone, A. Paviglianiti, P. Frontera, and F. C. Morabito, *Toward an Automatic Classification of SEM Images of Nanomaterials via a Deep Learning Approach*. Singapore: Springer Singapore, 2020, pp. 61–72. [Online]. Available: https://doi.org/10.1007/978-981-13-8950-4_7 43

[66] C. Ieracitano, A. Paviglianiti, M. Campolo, A. Hussain, E. Pasero, and F. C. Morabito, "A novel automatic classification system based on hybrid unsupervised and supervised machine learning for electrospun nanofibers," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 1, pp. 64–76, 2020. 44

[67] C. Garbin, X. Zhu, and O. Marques, "Dropout vs. batch normalization: an empirical study of their impact to deep learning," *Multimedia Tools and Applications*, vol. 79, pp. 12 777–12 815, 2020. 44

[68] S. Loussaief and A. Abdelkrim, "Convolutional neural network hyper-parameters optimization based on genetic algorithms," *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 10, 2018. 45

[69] A. Akl, I. El-Henawy, A. Salah, and K. Li, "Optimizing deep neural networks hyperparameter positions and values," *Journal of Intelligent & Fuzzy Systems*, vol. 37, no. 5, pp. 6665–6681, 2019. 45

[70] H. Li, P. Chaudhari, H. Yang, M. Lam, A. Ravichandran, R. Bhotika, and S. Soatto, "Rethinking the hyperparameters for fine-tuning," *arXiv preprint arXiv:2002.11770*, 2020. 46

[71] D. Pedamonti, "Comparison of non-linear activation functions for deep neural networks on MNIST classification task," *CoRR*, vol. abs/1804.02763, 2018. [Online]. Available: http://arxiv.org/abs/1804.02763 46

[72] W.-C. Yeh, Y.-P. Lin, Y.-C. Liang, C.-M. Lai, and C.-L. Huang, "Simplified swarm optimization for hyperparameters of convolutional neural networks," *Computers & Industrial Engineering*, vol. 177, p. 109076, 2023. 46

[73] E. Tuba, N. Bačanin, I. Strumberger, and M. Tuba, "Convolutional neural networks hyperparameters tuning," in *Artificial intelligence: theory and applications*. Springer, 2021, pp. 65–84. 46

[74] I. Cardoza, J. P. García-Vázquez, A. Díaz-Ramírez, and V. Quintero-Rosas, "Convolutional neural networks hyperparameter tunning for classifying firearms on images," *Applied Artificial Intelligence*, vol. 36, no. 1, p. 2058165, 2022. 46

[75] J.-E. Lee, Y.-B. Kim, and J.-N. Kim, "Hyperparameter optimization for image classification in convolutional neural network," *Journal of the Institute of Convergence Signal Processing*, vol. 21, no. 3, pp. 148–153, 2020. 47

[76] D. Marinov and D. Karapetyan, "Hyperparameter optimisation with early termination of poor performers," in *2019 11th Computer Science and Electronic Engineering (CEEC)*. IEEE, 2019, pp. 160–163. 47

[77] H. Alibrahim and S. A. Ludwig, "Hyperparameter optimization: Comparing genetic algorithm against grid search and bayesian optimization," in *2021 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2021, pp. 1551–1559. 47

[78] S. Mekruksavanich, P. Jantawong, N. Hnoohom, and A. Jitpattanakul, "Hyperparameter tuning in convolutional neural network for face touching activity recognition using accelerometer data," in *2022 Research, Invention, and Innovation Congress: Innovative Electricals and Electronics (RI2C)*. IEEE, 2022, pp. 101–105. 47

[79] R. Gonzales-Martinez, J. Machacuay, P. Rotta, and C. Chinguel, "Hyperparameters tuning of faster r-cnn deep learning transfer for persistent object detection in radar images," *IEEE Latin America Transactions*, vol. 20, no. 4, pp. 677–685, 2022. 48

[80] J.-H. Yoo, H.-i. Yoon, H.-G. Kim, H.-S. Yoon, and S.-S. Han, "Optimization of hyper-parameter for cnn model using genetic algorithm," in *2019 1st International conference on electrical, control and instrumentation engineering (ICECIE)*. IEEE, 2019, pp. 1–6. 49

[81] Y. LeCun, C. Cortes, and C. J. Burges, "Mnist handwritten digit database," http://yann.lecun.com/exdb/mnist/, 2010, accessed: 2024-12-13. 49

[82] S. A. Kawa and M. ArifWani, "Randomized search on a grid of cnn networks with simplified search space," in *2021 8th International Conference on Computing for Sustainable Global Development (INDIACom)*. IEEE, 2021, pp. 68–73. 49

[83] F. Talaat and S. Gamel, "Rl based hyper-parameters optimization algorithm (roa) for convolutional neural network. j ambient intell human comput (2022)." 50

[84] A. Krizhevsky, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009, technical Report. [Online]. Available: https://www.cs.toronto.edu/~kriz/cifar.html 50

[85] D. Fontes, "Nanoparticles images," https://data.mendeley.com/datasets/jb25t93n47/1, 2017, accessed: 2017-2-2. 54

[86] Y. Li, H. Zhang, X. Xue, Y. Jiang, and Q. Shen, "Deep learning for remote sensing image classification: A survey," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 8, no. 6, p. e1264, 2018. 56

[87] N. K. R. Mirayanti, S. Sariyasa, and I. G. A. Gunadi, "Batch size and learning rate effect in covid-19 classification using cnn," *Jurnal Mantik*, vol. 7, no. 3, pp. 1752–1765, 2023. 59

[88] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," *arXiv preprint arXiv:1609.04836*, 2016. 59

[89] Y. Bengio, "Practical recommendations for gradient-based training of deep architectures," 2012. [Online]. Available: https://arxiv.org/abs/1206.5533 60