UNIVERSIDAD TECNOLÓGICA DE LA MIXTECA

DIVISIÓN DE ESTUDIOS DE POSGRADO

METAHEURÍSTICAS PARA BÚSQUEDA DE HIPERPARÁMETROS DE UNA RED NEURONAL PROFUNDA APLICADA A CLASIFICACIÓN DE COVID-19

TESIS

PARA OBTENER EL TÍTULO DE MAESTRO EN TECNOLOGÍAS DE CÓMPUTO APLICADO

PRESENTA:
ING. NÉSTOR URIEL HERNÁNDEZ CORTEZ

DIRECTOR DE TESIS:
DR. RAÚL CRUZ BARBOSA

CO-DIRECTOR DE TESIS:
DR. ARTURO TÉLLEZ VELÁZQUEZ

HUAJUAPAN DE LEÓN OAXACA, ENERO DE 2024

Agradecimientos

A mis padres, Gabriela Cortez y Mario Hernández, por todo el cariño y apoyo que me han demostrado y todos los sacrificios que han realizado para permitirme estar aquí. A mi hermana, María de la luz, por ser siempre mi apoyo emocional y brindarme en todo momento su confianza. A mi hermano, Eli Emmanuel, por ser mi inspiración y principal motivo para seguir adelante. A mi familia, por enseñarme cada día el valor de la perseverancia, esfuerzo y dedicación.

A mi director y codirector de tesis, Dr. Raúl Cruz Barbosa y Dr. Arturo Téllez Velázquez, por su guía y consejos.

A mis sinodales: Dr. Ignacio Arroyo Fernández, Dr. Christian Eduardo Millán Hernández, Dr. Eduardo Sánchez Soto y Dr. Antonio Orantes Molina..

A la Universidad Tecnológica de la Mixteca (UTM) por brindarme la oportunidad de continuar con mi formación académica.

Al Consejo Nacional de Humanidades, Ciencias y Tecnologías (CONAHCyT), por el apoyo económico recibido a lo largo de este proyecto de tesis.

Índice general

Índice de figuras

Resumen

El COVID-19 es una enfermedad que puede llegar a causar severas complicaciones de salud y, en casos extremos, la muerte. El gran impacto que ha tenido a nivel mundial ha sido mayúsculo, debido a su rápida propagación y las dificultades que se han tenido para controlar el avance de esta enfermedad, aunado a los problemas que presentan los métodos de diagnóstico tradicionales como la prueba RT-PCR. Para afrontar estas dificultades, se han propuesto diferentes soluciones por medio de los sistemas de diagnóstico asistido por computadora, empleando desde clasificadores tradicionales hasta arquitecturas de aprendizaje profundo. Asimismo se ha comprobado que, tanto las imágenes de rayos X de tórax como las tomografías computarizadas, son de gran utilidad al momento de detectar la enfermedad. Son principalmente las arquitecturas de aprendizaje profundo las que han demostrado una efectividad y rendimiento satisfactorios; sin embargo, encontrar los valores para los hiperparámetros que aseguren un rendimiento óptimo es una tarea muy costosa, si este proceso se realiza de forma manual.

Debido a lo anterior, en el presente trabajo de tesis se realizan dos implementaciones de clasificación de COVID-19 basados en redes neuronales profundas utilizando imágenes de rayos X de tórax. En el primero de éstos, se utiliza de forma predeterminada la arquitectura *Resnext50_32x4d* para realizar únicamente la optimización de hiperparámetros de entrenamiento. La segunda implementación, realiza la optimización de hiperparámetros de estructura, generando diferentes estructuras CNN a partir de tres enfoques propuestos: modelo convolucional básico, modelo MFL y modelo residual. Para ambos casos, la optimización de los hiperparámetros se realiza mediante el uso de metaheurísticas, experimentando con la evolución diferencial, el algoritmo genético y lobos grises. Los resultados experimentales obtenidos muestran que, en ambas implementaciones se presenta una mejora en el rendimiento de clasificación en comparación con los modelos sin optimizar. Además, la optimización de hiperparámetros de entrenamientos funciona mejor, ya que mediante el uso de la evolución diferencial, se logró alcanzar un rendimiento general de 90.32 %, tanto en la medida de exactitud de clasificación como en sensibilidad sobre el conjunto de prueba de una base de datos curada de COVID-19.

Publicación derivada

Se obtuvo una publicación derivada a partir del presente trabajo de tesis, la cual es la siguiente:

- Hernández-Cortez, N.U., Téllez-Velázquez, A., Cruz-Barbosa, R. Sobre la importancia de los datos curados de COVID-19 para mejorar la explicación visual de Grad-CAM. Workshop on Explainable Deep Learning for COVID-19 data, IBERAMIA 2022, Cartagena, Colombia.

En la publicación se presentan los resultados de un método explicativo (*post hoc*), basado en la técnica Grad-CAM, que permite realizar un análisis visual sobre la importancia de la calidad de los datos y las características que conducen a la decisión de clasificación de la enfermedad de COVID-19. Se detalla la implementación de dos modelos de clasificación, sobre un conjunto de datos curado y otro no-curado. Adicionalmente, se presenta un análisis visual a *posteriori* sobre los patrones que ha aprendido cada modelo y su relevancia para la clasificación.

Capítulo 1

Introducción

La enfermedad de COVID-19 es causada por el virus del síndrome respiratorio agudo severo coronavirus 2 (SARS-CoV-2, por sus siglas en inglés), perteneciente a la familia de los coronavirus. Entre las principales razones de su rápida propagación se encuentra su corto período de incubación, el cual se estima entre 1 y 14 días con una media que varía de entre 5.6 a 6.7 días (Quesada et al., 2021). Esto ha causado que las personas infectadas se conviertan en la principal fuente de propagación del virus (Calvo et al., 2020). Esta enfermedad actúa principalmente deteriorando el sistema respiratorio de manera progresiva. El sistema cardiovascular también se ve afectado haciendo a las personas con enfermedades cardiovasculares más propensas a una descompensación que puede incluso causar la muerte, en el peor de los casos (López-Ponce de León et al., 2020). Como ejemplo de las afectaciones, en el año 2021, las cifras de casos confirmados de COVID-19 a nivel mundial superaron los 695 millones, mientras que el número de decesos se acerca a los 7 millones, convirtiéndose de esta manera en una de las más grandes pandemias de los últimos siglos (worldometers, 2021). En el caso de México, el impacto fue considerable, pues tan solo para el año 2020, el COVID-19 llegó a posicionarse como la segunda causa de muerte a nivel nacional y manteniéndose entre las cinco principales para el año 2022 (INEGI, 2021, 2023).

Para la detección de esta enfermedad el método más utilizado es la prueba de la Reacción en Cadena a la Polimerasa en Tiempo Real (RT-PCR, por sus siglas en inglés), la cual a pesar de ser confiable puede llegar a presentar algunas falencias (Corman et al., 2020). Por lo anteriormente mencionado, buscar una mejora en las formas de detección de esta enfermedad representa un papel primordial en la detección temprana y oportuna para evitar que la enfermedad se propague. Un software de detección automática puede desempeñar un papel importante al momento de auxiliar en el diagnóstico de COVID-19, corroborando la decisión del experto y funcionando como medio de referencia en situaciones de incertidumbre. Sin embargo, dada la reciente aparición de la enfermedad, la caracterización de ésta y la construcción de sistemas capaces de realizar una correcta clasificación aún continúan en experimentación, estudiando diferentes enfoques para afrontar esta problemática. Por ejemplo, en Iwendi et al. (2021) se aborda desde el uso de clasificadores tradicionales, mientras

que en trabajos como Lopez-Betancur et al. (2021) se estudia el comportamiento de diferentes modelos de Redes Neuronales Convolucionales (CNNs, por sus siglas en inglés). En este mismo sentido, también se ha explorado el comportamiento utilizando modelos de CNNs pre-entrenadas, tal como se observa en Heidari et al. (2020) y en Khan and Aslam (2020). Una de las mejores opciones para realizar este tipo de trabajos es mediante el análisis de imágenes de Tomografías Computarizadas (CT, por sus siglas en inglés) o de Rayos X (XR, por sus siglas en inglés). Por un lado, las imágenes de CT suelen aportar información adicional que puede ayudar a incrementar considerablemente la sensibilidad de la prueba RT-PCR; sin embargo, requiere de equipo especializado y es altamente costoso, lo cual reduce la cantidad de personas que pueden tener a su alcance dicho estudio. Las imágenes de XR pueden ayudar a mejorar la sensibilidad de la prueba RT-PCR, además de que son mucho más baratas y pueden estar al alcance de la mayoría de los hospitales del mundo, favoreciendo al uso de sistemas de diagnóstico asistido por computadora (Perez et al., 2021). Además, este tipo de imágenes ayuda a detectar de forma visual algunas anomalías características de la enfermedad de COVID-19, tales como, la *consolidación*, la *opacidad lineal* y la *opacidad de vidrio esmerilado* en la zona pulmonar (Cleverley et al., 2020).

Un sistema de diagnóstico asistido por computadora sería una herramienta que puede ser de mucha utilidad al momento de realizar diagnósticos mediante el uso de imágenes médicas, ya que proporcionan un segundo punto de vista al radiólogo de una manera más rápida y segura (Karar et al., 2021). En este contexto, las arquitecturas de aprendizaje profundo han demostrado tener buen rendimiento al momento de realizar clasificación de imágenes, gracias a su capacidad de aprender patrones y características en múltiples niveles de abstracción (LeCun et al., 2015). El funcionamiento de estas arquitecturas puede mejorar haciendo una correcta elección de los valores de los hiperparámetros de la Red Neuronal Profunda (DNN, por sus siglas en inglés), ya que estos influyen de manera directa en el rendimiento. Sin embargo, resulta muy complicado seleccionar manualmente aquellos que generen resultados óptimos, por lo que es recomendable obtenerlos mediante algún algoritmo de ajuste automático (Probst et al., 2019).

El presente proyecto de tesis tiene como objetivo optimizar los hiperparámetros de estructura y de entrenamiento de redes DNN novedosas mediante técnicas de optimización metaheurísticas para la clasificación de COVID-19 usando imágenes de XR de tórax de dos bases de datos conocidas (Chest+COVID-19 y COVIDGR).

Para lograr ésto, se realiza el entrenamiento de una DNN, centrándose en la optimización de los hiperparámetros mediante metaheurísticas, cuya optimización se aborda a partir de dos implementaciones diferentes. En el primero, se utiliza una arquitectura CNN predefinida para realizar la optimización únicamente de hiperparámetros de entrenamiento. En el segundo, se realiza la optimización principalmente de hiperparámetros de estructura, experimentando con tres enfoques para la construcción de la red: capas convolucionales básicas, bloques multiescala y bloques residuales. Finalmente, la clasificación de imágenes se realiza sobre un conjunto de datos homogéneo conocido como COVIDGR. Dicho conjunto destaca por tener

una construcción selectiva en la que participaron cuatro expertos, además de evitar algunas de las falencias típicas de los conjuntos de datos tradicionales de COVID-19, tales como datos sesgados a casos de gravedad y señalizaciones de imagen o elementos ajenos al cuerpo del paciente tanto fuera como dentro de la zona pulmonar.

1.1. Planteamiento del problema

A finales del 2019, en la ciudad de Wuhan, China, se detectaron los primeros casos de lo que parecía ser un nuevo tipo de neumonía vírica, la cual era causada por un nuevo tipo de coronavirus denominado SARS-CoV-2 o también conocido como COVID-19. Esta enfermedad comenzó a esparcirse por todo el mundo y, finalmente, fue declarada como emergencia sanitaria a escala global el 30 de enero del 2020 (WHO, 2021). La principal fuente de la propagación del SARS-CoV-2 son las mismas personas infectadas; su contagio se realiza al momento que las partículas suspendidas en el aire, expulsadas por los infectados al momento de hablar o estornudar, entran en contacto con la nariz, boca, ojos o alguna mucosa de alguna otra persona sana (Calvo et al., 2020). Actualmente, los casos confirmados de COVID-19 a nivel mundial superan la cifra de los 245 millones, mientras que el número de decesos está cerca de los 5 millones (worldometers, 2021). En lo que respecta a México, se tienen registros de un total de 3,793,783 casos confirmados y 287,274 defunciones (GOBIERNO DE MÉXICO, 2023), cifras que se incrementan diariamente, ya que la pandemia sigue activa.

Dada la alta tasa de contagios de esta enfermedad y los peligros que representa para la salud, el no ser tratada de manera oportuna resulta importante. Tradicionalmente, la detección del COVID-19 se realiza mediante la prueba RT-PCR, la cual se encarga de detectar el ácido ribonucleico (ARN) mensajero que contiene la presencia de SARS-CoV-2 (Corman et al., 2020). Aunque este método es eficiente, también presenta algunas falencias, entre las que se encuentran: la baja disponibilidad de insumos en algunas regiones, los altos costos y la necesidad de equipo y personal especializado, además de posibles errores humanos. Es por esto que las modalidades de imágenes médicas, como las XR de tórax o las CT, se presentan como una herramienta útil para el diagnóstico de esta enfermedad (Kovács et al., 2021).

La enfermedad de COVID-19 suele causar afecciones en los pulmones, las cuales se reflejan en características particulares (o anormales) que son detectables visualmente en imágenes radiológicas (Guo et al., 2020). Estas características en las imágenes pueden ser aprovechadas por las técnicas de visión computacional para realizar la clasificación de esta enfermedad. En la literatura se pueden encontrar ejemplos del uso de imágenes médicas para la clasificación de COVID-19, abordado este tema desde diferentes enfoques (Jangam et al., 2021; Punn and Agarwal, 2021; Lopez-Betancur et al., 2021; Abbas et al., 2021; Wang et al., 2020). Como se puede observar, el uso de las DNNs es uno de los enfoques más utilizados y esto se debe a que suelen mostrar resultados satisfactorios. Sin embargo, uno de los principales inconvenientes, al momento de trabajar con estas redes, es el ajuste de los hiperparámetros como la tasa de aprendizaje (LR, por sus siglas en inglés), el tamaño de lote (BS, por sus siglas en inglés),

el número de épocas, la constante de impulso, el coeficiente de regularización, el número y tamaño de capas convolucionales, el tamaño del kernel, entre otros. La importancia de realizar un correcto ajuste de hiperparámetros radica en que estos influyen de manera directa en el rendimiento de clasificación. El problema surge cuando el ajuste de hiperparámetros se realiza de forma manual, ya que es una tarea tediosa debido a la gran cantidad de combinaciones que pueden presentarse, lo cual se traduce en un gran costo computacional. Para atenuar este problema, se ha comenzado a usar metaheurísticas con la intención de optimizar la búsqueda de aquellas combinaciones que generen resultados óptimos en términos de diferentes medidas como la sensibilidad, exactitud, especificidad, F1-score, entre otras. Un ejemplo es el caso de Goel et al. (2021) donde se realiza la clasificación de COVID-19 sobre imágenes XR de tórax por medio de una CNN optimizando los hiperparámetros mediante la técnica de manada de lobos grises. En dicho trabajo, se obtuvo una mejora considerable en el rendimiento de clasificación respecto al mismo modelo de red sin utilizar la metaheurística mencionada.

El presente proyecto de tesis tiene como objetivo optimizar los hiperparámetros de estructura y de entrenamiento de redes DNN novedosas mediante técnicas de optimización metaheurísticas para la clasificación de COVID-19 usando imágenes de XR de tórax de dos bases de datos conocidas (Chest+COVID-19 y COVIDGR). La atención se centra en la optimización de los hiperparámetros de las redes neuronales mediante metaheurísticas como son los algoritmos evolutivos o de manadas, con la finalidad de obtener resultados de rendimiento altos (mayores a 90 % de sensibilidad, exactitud o F1-score) en la tarea de clasificación.

1.2. Justificación

La enfermedad de SARS-CoV-2 o COVID-19 se ha convertido en una problemática a nivel mundial que ha afectado en gran medida aspectos económicos, sociales, medioambientales, entre otros, inclusive parece haber cambiado de forma permanente el concepto que se tenía sobre diversos aspectos de la vida cotidiana (Zaccato, 2020). Uno de los sectores sociales mayormente afectados son las personas pertenecientes a un estatus socio-económico bajo, ya que regularmente suelen presentar peores condiciones de salud debido a su particular estilo de vida y dificultades para el acceso a los servicios de salud. Debido a las comorbilidades y a la dificultad para mantener el confinamiento, la probabilidad de contagio de esta enfermedad se incrementa (Sánchez-Rivas, 2020). En México, el impacto de esta enfermedad ha sido mayúsculo, ya que tan sólo en el año 2020, el coronavirus se posicionó como la segunda causa de muerte, justo debajo de las enfermedades cardiacas (INEGI, 2021). Una de las principales dificultades para prevenir contagios es la determinación de la enfermedad por COVID-19 mediante la prueba RT-PCR. Debido a las falencias que esta prueba puede llegar a presentar, las modalidades de imágenes médicas resultan una buena alternativa como un segundo punto de vista (Kovács et al., 2021). Además, como se trata de una enfermedad reciente, todavía no se ha terminado de caracterizarla, por lo que la carencia de pruebas que permitan diagnosticar el COVID-19 de manera rápida sigue siendo un problema, pues esto

impide prevenir la enfermedad oportunamente.

Aun cuando el diagnóstico del COVID-19 ha sido abordado desde distintos enfoques por la comunidad científica de la inteligencia artificial, por ejemplo, en Iwendi et al. (2021) se analiza el rendimiento de los árboles de decisión (DT, por sus siglas en inglés), k-vecinos más cercanos (KNN, por sus siglas en inglés) y las Máquinas de Soporte Vectorial (SVM, por sus siglas en inglés), los cuales obtuvieron resultados satisfactorios; particularmente en el campo de las DNNs, se han obtenido muy buenos resultados, tal es el caso de Abbas et al. (2021) donde se utiliza un modelo de CNN conocido como Decompose, Transfer, and Compose (DeTraC), para la clasificación. Otro caso de éxito es de Perrilliat-García et al. (2020) donde también se usa un modelo de CNN; sin embargo, el ajuste de los hiperparámetros sigue siendo un problema latente. Por lo anterior, se siguen explorando nuevas arquitecturas y técnicas que ayudan a mejorar el desempeño en el diagnóstico; ejemplo de esto es la optimización a través de metaheurísticas como en Goel et al. (2021).

Para impactar en la problemática expuesta en este trabajo de tesis, se propone desarrollar un módulo de clasificación basado en una DNN optimizada, con la finalidad de ayudar a realizar la detección usando imágenes XR de tórax. Esto es, se pretende generar un modelo de clasificación de COVID-19 que presente alto rendimiento para la asistencia en el diagnóstico de dicha enfermedad, la selección de los hiperperámetros de dicho modelo se realizara mediante optimización metaheurística, evitando la complejidad y esfuerzo de un ajuste manual.

El presente trabajo de tesis es viable respecto a recursos computacionales, ya que para la experimentación correspondiente con métodos de DNNs, metaheurísticas y procesamiento de imágenes, la Universidad Tecnológica de la Mixteca cuenta con la infraestructura tales como servidores o cluster computacional necesarios para el cómputo intensivo o de alto rendimiento respectivo.

1.3. Hipótesis

Los hiperparámetros de una red neuronal profunda encontrados mediante métodos de optimización metaheurística incrementan el rendimiento de clasificación de COVID-19 utilizando imágenes médicas de XR de tórax.

1.4. Objetivos

1.4.1. Objetivo general

Implementar un algoritmo basado en una red neuronal profunda optimizada mediante búsqueda metaheurística de sus hiperparámetros, de tal forma que ayude a mejorar el desempeño de clasificación de COVID-19.

1.4.2. Objetivos específicos

1. Revisar el estado del arte sobre los trabajos relacionados con la clasificación de COVID-19 basada en redes neuronales profundas, así como, métodos de optimización de hiperparámetros de dichas redes mediante metaheurísticas.

2. Seleccionar un conjunto de datos público utilizado para clasificación de COVID-19.

3. Seleccionar los métodos de preprocesamiento de imágenes que aporten un mayor realce a las características principales para el diagnóstico de COVID-19.

4. Implementar un algoritmo que utilice una red neuronal profunda para realizar la clasificación de COVID-19.

5. Implementar al menos tres metaheurísticas que permitan optimizar los hiperparámetros de la red neuronal profunda del objetivo 4 que ayude en la mejora de su rendimiento.

6. Comparar el rendimiento del modelo de red optimizado (encontrado en el objetivo 5), en cuanto a clasificación de COVID-19 se refiere con otros modelos de red optimizados encontrados en el estado del arte.

1.5. Metas

1. Elaboración de un reporte sobre las metaheurísticas utilizadas para la optimización de los hiperparámetros de una red neuronal profunda en la clasificación de COVID-19.

2. Implementación de, al menos, tres metaheurísticas para optimización de parámetros.

3. Implementación de un algoritmo de clasificación de COVID-19 basado en una red neuronal profunda optimizada mediante alguna de las metaheurísticas de la meta anterior.

4. Creación de una biblioteca en lenguaje Python, para la ejecución de los algoritmos seleccionados.

5. Elaboración de un cuadro comparativo de rendimiento del modelo de red neuronal profunda con o sin uso de metaheurísticas.

6. Publicación de un artículo.

1.6. Trabajo relacionado

Debido a que la enfermedad de COVID-19 es reciente, aún no se ha completado su caracterización, por lo que el problema de su clasificación ha sido abordado desde diferentes enfoques, como el asistido por computadora, en el que se utilizan imágenes médicas para la detección. En el caso de las imágenes XR de tórax y de las CT, el objetivo consiste en detectar las características correspondientes a la enfermedad presentes en regiones de la zona pulmonar.

Los trabajos tan diversos sobre la clasificación de COVID-19 han propiciado que se puedan encontrar ejemplos tanto de técnicas poco especializadas, como es el caso de los clasificadores tradicionales, como de otras más sofisticadas, como los enfoques de aprendizaje profundo. Estos últimos, han demostrado ser eficientes para resolver problemas de diferentes campos del conocimiento. Ejemplos de estos los podemos encontrar en el área de los sistemas de reconocimiento de voz con el trabajo de Hinton et al. (2012), el reconocimiento de dígitos escritos a mano con el trabajo de LeCun et al. (1998) y principalmente en el área de clasificación de imágenes donde las CNNs han demostrado ser eficaces para esta tarea (Krizhevsky et al., 2012). Los enfoques de aprendizaje profundo se pueden caracterizar entre aquellos que utilizan modelos pre-entrenados con aprendizaje por transferencia o *Transfer Learning* (TL) y los que se enfocan en desarrollar técnicas personalizadas con la finalidad de superar el rendimiento de clasificación respecto a los modelos existentes (Islam et al., 2021).

Los enfoques de clasificadores tradicionales suelen ofrecer una implementación más sencilla en comparación con los enfoques de aprendizaje profundo. Una comparación de estas técnicas, en cuanto a clasificación de COVID-19, se puede encontrar en Iwendi et al. (2021), donde se analiza el rendimiento de los árboles de decisión, k-vecinos más cercanos y las máquinas de soporte vectorial (SVM, por sus siglas en inglés). Aunque este tipo de clasificadores han presentado resultados satisfactorios, aún son poco aptos para ser utilizados en el ámbito médico como un clasificador confiable. Con la finalidad de mejorar estos resultados, se ha optado por utilizar estos clasificadores en combinación con técnicas más sofisticadas. Por ejemplo, en el trabajo de Ismael and Şengür (2021), se realiza una comparación entre el rendimiento de una SVM que utiliza características obtenidas por diferentes modelos de CNNs y una CNN creada por los autores. A pesar de que los resultados obtenidos por las máquinas de soporte vectorial mejoran, estos son inferiores a los presentados por el modelo de CNN desarrollado.

Por otro lado, los enfoques de aprendizaje profundo suelen presentar una gran efectividad al momento de realizar la detección y clasificación de COVID-19 (Chandra et al., 2021; Yoo et al., 2020; Khuzani et al., 2021; Chowdhury et al., 2020). Uno de los métodos más utilizados es el uso de modelos pre-entrenados mediante TL. Esto lo podemos observar claramente en el trabajo de Lopez-Betancur et al. (2021), donde se realiza la comparación de 32 arquitecturas de CNNs para la detección de COVID-19 sobre una base de datos desbalanceada, obteniendo como resultado que, incluso la red menos eficaz supera a los clasificadores tradicionales. De

igual manera, en Punn and Agarwal (2021) se utilizan modelos de CNNs pre-entrenados para la clasificación de COVID-19 y la atención se centra en las técnicas para evitar el problema de la base de datos desbalanceada. Mientras que otros trabajos se centran en el preprocesamiento de los datos, como es el caso de Singh et al. (2021), quienes proponen realizar segmentación de los pulmones en las imágenes XR de tórax para realizar el entrenamiento de las CNNs. Finalmente, en Jangam et al. (2021) se usa el apilamiento (*Stacking*) de los modelos pre-entrenados para tratar de obtener mejores resultados.

Al igual que se utilizan diferentes técnicas en combinación con los modelos de CNNs pre-entrenados para mejorar el rendimiento de clasificación, también han surgido propuestas que introducen sus propios modelos de DNNs, con la finalidad de adaptarse de mejor manera al problema de clasificación de COVID-19. Tal es el caso de Abbas et al. (2021) quienes introdujeron su modelo conocido como descomposición, transferencia y composición (DeTraC) al problema de clasificación de COVID-19. En Mohimont et al. (2021) se propone un modelo de CNN clásico y una CNN temporal para predecir múltiples indicadores de COVID-19. Otro ejemplo es el de Wang et al. (2020), donde se propone un diseño de CNN para la detección de COVID-19 nombrada como COVID-Net, además de que presentan COVIDx, una de las bases de datos más grandes sobre COVID-19.

El uso de técnicas basadas en aprendizaje profundo han demostrado ser una buena alternativa para tratar el problema de clasificación de COVID-19, siendo las CNNs el principal recurso a utilizar debido a que suelen presentar los mejores resultados en el estado del arte. Sin embargo, este tipo de redes neuronales presenta la dificultad de realizar el ajuste de los parámetros tanto de estructura como de entrenamiento de la red, puesto que realizar esta acción de forma manual es tediosa y complicada debido a la gran cantidad de combinaciones que pueden presentarse.

La optimización de los hiperparámetros usando metaheurísticas ha demostrado ser una técnica que puede mejorar el rendimiento de las DNN, cuestión que se puede corroborar en los distintos trabajos de investigación en los que se ha implementado. En Serizawa and Fujita (2020) realizan la optimización de los hiperparámetros de una CNN por medio de un algoritmo de optimización de enjambre de partículas, obteniendo una mejora de hasta un 4.9 % en la exactitud de clasificación. Por otro lado, en Rodríguez-Hernández and Ochoa-Domínguez (2021) utilizaron métodos de optimización Bayesiana, logrando incrementar el número de conteos válidos en sinogramas tridimensionales de tomografías reduciendo la pérdida de la validación al momento de entrenar a la red. En el caso de la clasificación de COVID-19, para el trabajo de Goel et al. (2021) se usan metaheurísticas para encontrar las combinaciones de hiperparámetros que generen resultados óptimos, obteniendo resultados satisfactorios incluso con pocas épocas de entrenamiento.

1.7. Limitaciones

El presente trabajo de tesis se limita a realizar la clasificación de COVID-19 utilizando únicamente imágenes XR de tórax mediante una DNN. También se realiza la optimización de los hiperparámetros de dicha red utilizando las metaheurísticas de: Algoritmo Genético (GA, por sus siglas en inglés), Evolución Diferencial (DE, por sus siglas en inglés) y Optimización con Lobos Grises (GWO, por sus siglas en inglés). Los datos a utilizar para la experimentación corresponden a dos conjuntos de datos públicos (*Chest+COVID-19* y COVIDGR). Dichos conjuntos de datos contienen información de diversos sitios, tales como hospitales e instituciones de salud. Además, se contempla la realización de dos análisis comparativos sobre el rendimiento del modelo de red optimizada. El primero, con respecto a los resultados de la misma red sin optimizar; y el segundo, respecto a otros métodos de la literatura relacionados con metaheurísticas.

1.8. Metodología

Para abordar el tema de la clasificación de COVID-19 usando imágenes XR de tórax, se realizará un estudio sobre las características que presentan las imágenes de pacientes que padecen esta enfermedad, así como los diferentes métodos que se han utilizado en el ámbito computacional. Lo anterior, haciendo hincapié en el uso de redes neuronales profundas para mejorar el rendimiento de clasificación al realizar optimización mediante alguna metaheurística.

El desarrollo de este trabajo de tesis se llevará a cabo en tres etapas constituidas de la siguiente manera: tratamiento de los datos, implementación del clasificador y análisis comparativo. Estas tres etapas se presentan en la Fig. 1.1.

- Etapa 1. En esta etapa se recopilan y preparan los datos que serán utilizados para realizar la clasificación. Se divide en dos pasos:

    1. Adquisición de datos. La obtención de un conjunto de datos es un factor importante para la clasificación, es por esto que se realizará una selección de conjuntos de datos públicos compuestos por imágenes XR de tórax recopiladas de diferentes sitios (hospitales, instituciones de salud, etc.). Dichos conjuntos son: *Chest+COVID-19* y COVIDGR; los cuales ya han sido probados y utilizadas en trabajos de clasificación de COVID-19. Lo anterior permitirá realizar una comparación de la presente propuesta con la literatura relacionada.
    2. Preprocesamiento. Debido a que los datos se recopilarán de diversas fuentes, se espera que características como el tamaño y el ruido varíen entre las distintas bases. Por esta razón, se realizará un preprocesamiento con la finalidad de homogeneizar los datos, reducir la información irrelevante y resaltar las características principales que ayuden a mejorar la clasificación.

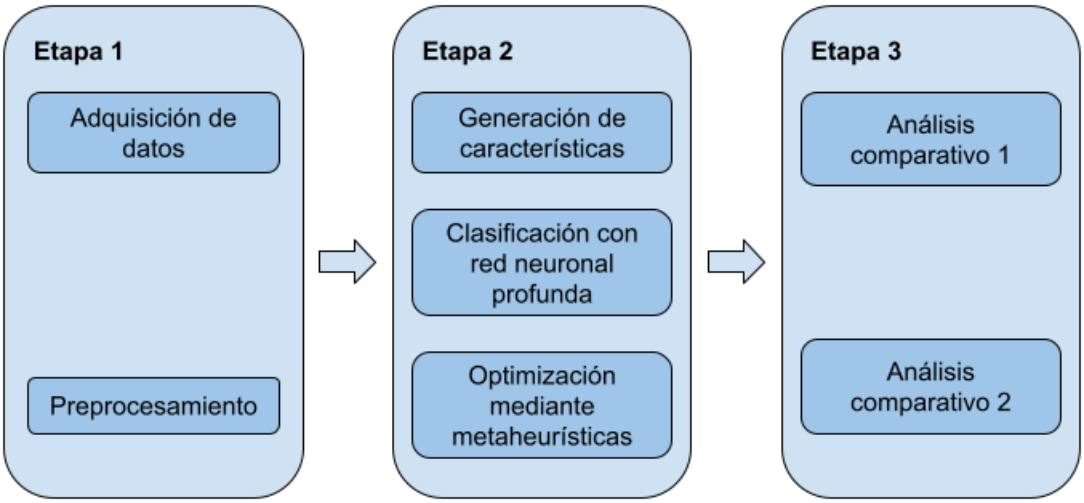


Figura 1.1: Metodología para realizar la clasificación de COVID-19

- Etapa 2. En esta etapa se realiza la implementación del algoritmo de clasificación, lo que incluye el entrenamiento y la optimización de la DNN seleccionada. Se divide en tres pasos:

  1. Generación de características. En este paso se pretende obtener un conjunto de características, presentes en las imágenes XR de tórax que favorezcan a la clasificación de COVID-19. Dichas características se pueden obtener de manera automática, para el caso de CNNs.
  2. Clasificación con red neuronal profunda. En este paso se realiza la implementación y entrenamiento de una DNN, utilizando las características extraídas por la misma red y estableciendo el valor de los hiperparámetros de la red neuronal de forma manual, es decir, sin realizar la optimización de los hiperparámetros.
  3. Optimización mediante metaheurísticas. Aquí se realiza la optimización de hiperparámetros de la DNN utilizada en el paso 2. Para esto, se experimenta con al menos tres metaheurísticas actuales de la literatura, seleccionando aquella que presente el mejor rendimiento.

- Etapa 3. En esta etapa se hace la evaluación y el análisis de los resultados obtenidos en los diferentes experimentos.

  1. Análisis comparativo 1. Se realizará un análisis comparativo de rendimiento del clasificador observando las diferencias que se presentan entre la red profunda con los hiperparámetros establecidos manualmente y de forma optimizada.
  2. Análisis comparativo 2. Se realiza una comparación de los resultados obtenidos en este trabajo con otros relacionados en la literatura.

Capítulo 2

Marco teórico

En este capítulo se presenta la base teórica que sustenta el presente proyecto de tesis. En la primera sección se presenta una introducción a la enfermedad de COVID-19, así como una descripción de imágenes XR de tórax, las cuales han sido utilizadas para clasificar pacientes con COVID-19. La segunda sección muestra los métodos de preprocesamiento de imágenes relevantes para el presente trabajo. En la tercera sección se hace una breve introducción a las DNNs utilizadas para esta tarea. Finalmente, en la última sección se presentan algunos métodos de optimización metaheurística que han sido usados para optimizar los hiperparámetros de DNNs.

2.1. Clasificación de COVID-19 mediante imágenes de rayos X

En esta sección se muestra una breve explicación del panorama actual de la enfermedad de COVID-19, tanto a nivel nacional como a nivel internacional. En el segundo punto se presenta una descripción de las imágenes XR, sus características principales y los hallazgos relacionados con dicha enfermedad.

2.1.1. COVID-19

La enfermedad de COVID-19 es causada por el virus SARS-CoV-2, la cual se caracteriza por causar infecciones respiratorias y síntomas similares a los de la gripe, a los cuales se le une, la dificultad para respirar y la pérdida del sentido del olfato y gusto (Pérez-Abreu et al., 2020). Esta enfermedad afecta principalmente al sistema respiratorio de manera progresiva, pero también se ha encontrado que puede llegar a comprometer el sistema cardiovascular causando lesiones miocárdicas y una descompensación en enfermedades cardiovasculares. En los casos más graves, puede llegar a ocasionar síndrome de dificultad respiratoria aguda, choque séptico e incluso la muerte (López-Ponce de León et al., 2020).

La alta tasa de propagación de esta enfermedad se debe principalmente a que, según la Organización Mundial de la Salud (OMS), el período de incubación del virus se sitúa entre 1 y 14 días, considerando un periodo medio que varía de entre 5.6 a 6.7 días (Quesada et al., 2021). El contagio de esta enfermedad se da debido a las partículas suspendidas en el aire expulsadas por las personas infectadas al momento de hablar o estornudar, así como a través del contacto del virus con la boca, ojos, fosas nasales, o alguna mucosa de alguna persona sana. De esta manera, una persona contagiada puede no presentar síntomas de manera inmediata e inclusive puede cursar con la enfermedad de manera asintomática. (Calvo et al., 2020) menciona que la principal fuente de propagación son las mismas personas contagiadas.

Según la propuesta realizada por (Siddiqi and Mehra, 2020), la progresión de esta enfermedad en un paciente infectado puede dividirse en tres etapas.

- Etapa 1 o también conocida como *infección temprana*, se da en el momento del contagio y comprende el periodo de incubación del virus presentando síntomas respiratorios leves, malestar general, fiebre y tos seca. Internamente, es en esta etapa cuando el virus del SARS-CoV-2 se multiplica y se establece en el cuerpo del huésped.

- En la etapa 2, la enfermedad ya se encuentra establecida llegando a desarrollar una neumonía viral con tos, fiebre y una posible hipoxia (la cual se define como la ausencia de oxígeno suficiente en los tejidos como para mantener las funciones corporales). Debido a este conjunto de complicaciones causadas por esta enfermedad, la mayoría de pacientes necesita ser hospitalizado y debido a la neumonía viral causada por este virus, las imágenes XR de tórax y CT presentan características visuales, tales como la opacidad de vidrio esmerilado o la consolidación en la zona pulmonar.

- La etapa 3 se manifiesta como un *síndrome de hiperinflación sistémica extrapulmonar* y es la etapa de más riesgo, pues puede desembocar en una insuficiencia respiratoria o en en un colapso cardiovascular, llegando a afectar algunos órganos sistémicos, tales como el hígado y los riñones. Realmente, sólo una minoría de pacientes llega hasta esta etapa, pero si no recibe atención médica especializada inmediatamente puede llegar a la muerte.

La enfermedad de COVID-19 llegó a convertirse en una problemática a nivel mundial, no sólo por sus afecciones a la salud, sino también por las graves consecuencias en la economía, las relaciones sociales y problemas medio-ambientales; todo esto debido a la cuarentena impuesta por todos los gobiernos del mundo (Zaccato, 2020). Al momento de la redacción de este trabajo de tesis, los casos confirmados de COVID-19 a nivel mundial superan los 245 millones de personas, mientras que el número de decesos está cerca de los 5 millones (worldometers, 2021). En lo que respecta a México, se tienen registros de un total de 3.7 millones casos confirmados y 287274 defunciones (GOBIERNO DE MÉXICO, 2023). El impacto en este país ha sido considerable, ya que tan sólo en el año 2020, el coronavirus

se posicionó como la segunda causa de muerte, justo debajo de las enfermedades cardíacas, registrando un 18.4 % de las muertes totales (INEGI, 2021).

2.1.2. Uso de imágenes de rayos X de tórax

El principal método de detección del COVID-19 es la prueba RT-PCR, la cual ha demostrado ser efectiva para detectar el material genético de virus SARS-CoV-2 en el ARN mensajero, mediante el estudio de muestras recolectadas del tracto respiratorio por medio del hisopado nasofaríngeo u orofaríngeo (Corman et al., 2020). Sin embargo, este método tiene una gran dificultad al momento de hacer la detección en un paciente con una baja carga viral, por lo que es necesario esperar un tiempo considerable para obtener resultados confiables. Aunado a esto, se presentan otros problemas como la dificultad en el acceso a estas pruebas, ya que a principios de esta pandemia, la escasez de los kits de extracción de ARN resultaron en unos de los mayores obstáculos en la obtención de pruebas de detección (Aguilar-Gamboa, 2020).

Dado que se ha demostrado clínicamente que los pacientes de COVID-19 presentan infección en los pulmones (Guo et al., 2020), las modalidades de imágenes radiológicas pueden ser consideradas un medio a través del cual realizar la detección de esta enfermedad. Las dos modalidades utilizadas para esto son las imágenes de CT y XR, en las cuales se puede encontrar características visuales de lesiones pulmonares provocadas por esta enfermedad.

Las CT son un tipo de imágenes que se obtiene por medio de una máquina de XR especial conocida como máquina de tomografía computarizada. En este tipo de imágenes, los XR que pasan a través del cuerpo de un paciente son recibidos por un sistema de detectores que los convierten en información. Dicha información es procesada y analizada por un sistema computacional para reconstruir una imagen digital, donde cada píxel es una representación de un volumen tridimensional en la trayectoria del haz de XR, obteniendo imágenes altamente detalladas de regiones del cuerpo. Gracias al nivel de detalle presentado en estas imágenes, las CT permiten definir y caracterizar de mejor manera las alteraciones causadas por la enfermedad de COVID-19, así como su distribución (Muñiz and Casanovas, 2006). Sin embargo, presentan algunos problemas, tales como: la baja disponibilidad del equipo necesario para obtener este tipo de imágenes, el alto peligro de contagio de esta enfermedad causada por la contaminación del túnel de CT y el peligro que representa la exposición prolongada de los pacientes a la radiación durante las CT.

En el caso de las imágenes XR de tórax convencionales, los XR que pasa a través del cuerpo de una persona quedan impresos en una lámina de poliéster especial y la imagen se forma gracias a la atenuación de los XR según la densidad de las estructuras del cuerpo que atraviesan. A diferencia de las CT, la digitalización de una imagen de XR convencional consiste únicamente en el escaneo de la placa radiográfica en lugar de la reconstrucción de la información obtenida por los XR, es por esto que presentan una menor resolución comparadas con las CT (Díaz et al., 2014). Sin embargo, las imágenes XR de tórax tienen una mayor

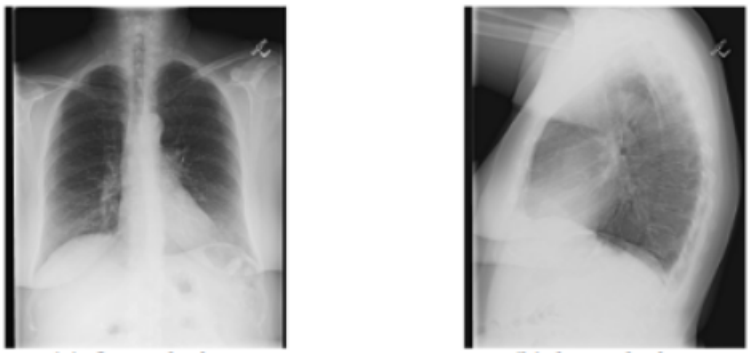


Figura 2.1: Radiografía de tórax de vista frontal y vista lateral. Fuente: (Xue et al., 2015)

disponibilidad en todo momento gracias a su uso común en la medicina, por lo que son más fáciles, rápidas y económicas de obtener. (Ghaderzadeh and Asadi, 2021), además de que el problema de la resolución se puede solventar con el uso de técnicas de aprendizaje profundo.

Por las anteriores características es que la mayoría de los trabajos relacionados con detección y diagnóstico de COVID-19 han optado por utilizar imágenes XR de tórax. Si bien, en un principio, las CT eran las preferidas por los investigadores, para el año 2020 el 57 % de los 160 trabajos revisados se enfocaron solamente en el uso de imágenes XR de tórax mientras que el 40 % siguió trabajando con CT y únicamente el 3 % utilizó ambas modalidades (Ghaderzadeh and Asadi, 2021).

Tal como se mencionó anteriormente, las radiografías de tórax son una técnica comúnmente usada para identificar anomalías en la caja torácica. Existen dos formas de capturar una XR de tórax (Xue et al., 2015):

- Vista frontal. Este tipo de vista suele dividirse a su vez en: posteroanterior, la cual es una proyección desde la espalda hacia la parte anterior del tórax; y anteroposterior, en donde la proyección atraviesa el tórax desde su parte anterior hasta la espalda.

- Vista lateral. Esta se realiza desde algunas de las laterales de la caja torácica y es utilizada para identificar lesiones que se encuentran detrás del corazón.

La Figura 2.1 muestra un ejemplo de estos dos tipos de vistas.

La enfermedad de COVID-19 suele presentar algunas características visuales en este tipo de imágenes (Manna et al., 2020):

- Las más comunes son la opacidades nebulosas u opacidades de vidrio esmerilado, las cuales suelen tener una morfología redondeada y se manifiestan principalmente en la periferia de las zonas pulmonares, siendo más comunes de encontrar en la parte inferior de las mismas. Estas opacidades pueden estar presentes en uno o ambos pulmones y, conforme la enfermedad avanza, pueden expandirse y aparecer en la zona media y, con menos frecuencia, en la parte superior de los pulmones.

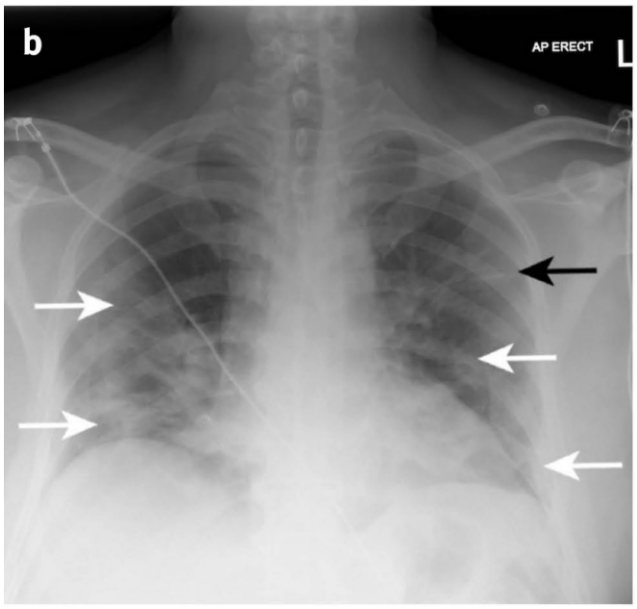


Figura 2.2: Radiografía de tórax de un paciente con COVID-19, la cual muestra opacidades de vidrio esmerilado (flechas blancas) y opacidad lineal (flecha negra). Fuente: (Cleverley et al., 2020)

- La consolidación es otro efecto causado por el crecimiento y la fusión de las opacidades de vidrio esmerilado provocado por el incremento en la densidad pulmonar, lo cual deriva en un incremento de la intensidad de los píxeles de la imagen que corresponden a las estructuras pulmonares.

- Otro efecto son las opacidades lineales, descritas como líneas blancas gruesas, periféricas y horizontales (Cleverley et al., 2020).

Estas características se pueden observar en la Figura 2.2.

Durante los primeros días de esta enfermedad, es común que las radiografías de tórax no muestren ningún tipo de alteraciones. Esto es debido a que, las afecciones presentadas aún son demasiado sutiles como para ser identificadas. Sin embargo, en el caso de aquellas que presentan algunas características visuales de la enfermedad, el hallazgo más frecuente son las pequeñas opacidades de vidrio esmerilado ubicadas en las zonas periféricas de los pulmones, principalmente en la zona inferior. Conforme la enfermedad avanza, estos pequeños parches de opacidades crecen hasta el punto de combinarse y agruparse, dando paso al efecto conocido como *consolidación*. Además de que, aparecen otras características como las opacidades lineales, éstas pueden encontrarse durante la segunda semana de la enfermedad, siendo alrededor del décimo día cuando se presenta el mayor grado de afección en los pulmones. En las etapas más avanzadas de la enfermedad, todas estas características se encuentran presentes

de manera más pronunciada con opacidades pulmonares que cubren la totalidad de la zona pulmonar en ambos pulmones (Gordo et al., 2021).

2.2. Métodos de preprocesamiento de imágenes

El preprocesamiento de las imágenes es una etapa importante para cualquier sistema de clasificación basado en imágenes, ya que ayuda a mejorar la calidad de la misma y a resaltar características importantes, proporcionando una imagen de mejor calidad que pueden ayudar a aumentar la eficacia de un clasificador.

Es un hecho común que, la calidad de las imágenes se vea afectada por la presencia de elementos indeseados e imperfecciones relacionadas con el método utilizado para la adquisición de las imágenes. Estas pueden llegar a afectar la exactitud en cualquier sistema de clasificación (Pal and Sudeep, 2016). Los defectos presentados en una imagen son variados y el tratamiento de éstas no es un trabajo simple, por lo que se debe poner especial atención a ésto. Se han propuesto en la literatura diferentes técnicas que ayudan a mejorar la información de una imagen para tener éxito en pasos posteriores. Dichas técnicas suelen clasificarse en tres tipos: los métodos basados en operaciones puntuales, locales y globales (Rodríguez-Morales and Sossa-Azuela, 2012).

Los métodos que se basan en operadores puntuales utilizan operaciones básicas que ejecutan una transformación sobre un píxel a la vez y esta suele repetirse o aplicarse a cada uno de los píxeles que conforman la imagen. De esta manera, se modifica el contenido de la imagen y el mejoramiento de la calidad de la misma depende de la transformación realizada al píxel de entrada. Este tipo de métodos tienen la particularidad de que, las operaciones matemáticas que utilizan suelen ser expresiones simples y, dado que se aplican únicamente a cada píxel de la imagen, la velocidad de ejecución es alta, al menos en comparación con otros tipos de métodos.

Los métodos basados en operaciones locales utilizan una *vecindad* conformada por los píxeles adyacentes al píxel de entrada. Regularmente estas vecindades suelen ser de tamaño $n \times n$ ya que la operación se centra en un píxel específico, el cual suele ubicarse en el centro de la vecindad. Debido a que el uso de una vecindad de píxeles otorga un mejor análisis de la distribución de los niveles de intensidad en una determinada zona de la imagen, estos métodos suelen caracterizarse por comportarse como *filtros* con la capacidad de suavizar o realzar ciertas características de la imagen. Estos filtros pueden dividirse en lineales y no lineales, siendo los primeros basados en máscaras numéricas de convolución, mientras que los segundos se basan en operaciones estadísticas sobre los valores de intensidad pertenecientes a la vecindad, como por ejemplo, los filtros de orden estadístico.

Por último, los métodos de preprocesamiento basados en operaciones globales son aquellos que dependen de los valores de intensidad de todos y cada uno de los píxeles de la imagen. El uso de todos los valores de intensidad de la imagen puede dar paso a técnicas más complejas. Debido a la complejidad de estos cálculos, este tipo de técnicas suele tener un gran

costo computacional. Pero no todos los métodos de preprocesamiento perteneciente a esta categoría comparten esta característica, pues algunos como el estiramiento y ecualización del histograma se realizan utilizando operaciones simples.

2.2.1. Preprocesamiento de imágenes de rayos X de tórax

En el ámbito médico, las imágenes radiológicas se han convertido en unas de las más usadas como un elemento de apoyo para el profesional de la medicina. Este tipo de imágenes se suelen capturar sobre una lámina de poliéster especial, y gracias al avance de la tecnología, ya es posible obtenerlas directamente de forma digital (Díaz et al., 2014). Sin embargo, producto de la naturaleza de estas imágenes y debido al método de adquisición, este tipo de imágenes pueden presentar algunos problemas de calidad. La presencia de ruido aditivo es uno de los problemas más comunes que se pueden encontrar y puede causar múltiples inconvenientes al momento de trabajar con la imagen. La variación del brillo también es un factor a tener en cuenta y éste ocurre por dos razones principales: la primera es la cantidad de rayos X aplicados al momento de realizar la imagen y la segunda es la diferencia del tamaño del cuerpo del paciente (Heidari et al., 2020). Además, no es extraño que este tipo de imágenes contengan alguna etiqueta de referencia utilizada en el ámbito médico, pero que para el procesamiento computacional resulta irrelevante e incluso perjudicial (Punn and Agarwal, 2021). Aunado a esto, es posible realizar un mejoramiento de la imagen para resaltar algunas características importantes, por medio de la mejora del contraste de la imagen y el resaltado de los bordes (Restrepo, 1998).

Realce por ecualización de histograma

La ecualización del histograma es una técnica que consiste en distribuir los píxeles de la imagen en todo el rango de intensidad. En el caso ideal, esta distribución debería ser uniforme, obteniendo el mismo número de píxeles para cada valor de intensidad en la imagen. Sin embargo, esto no es posible en la práctica, por lo que esta técnica se limita a estirar el histograma en todo el rango posible, transformando píxeles con valores de intensidad similar a otros valores diferentes separándolos en la medida de lo posible. Al aplicar esta técnica, se logra obtener un mayor contraste entre los elementos de la imagen resaltando aquellos detalles que antes no eran tan evidentes (Gonzalez and Woods, 2008). Para la ecualización del histograma estándar, es necesario obtener el histograma acumulado de la imagen de entrada y procesarlo utilizando la transformación $T(r_k)$ denotada por la Ecuación 2.1.

$$s_k = T(r_k) = (L-1)\sum_{j=0}^{k} p_r(r_j) = \frac{(L-1)}{NM}\sum_{j=0}^{k} n_j \tag{2.1}$$

Donde L es el número total de niveles de intensidad, MN es el total de píxeles de la imagen de entrada, $p_r(r_j)$ es el valor del histograma acumulado para el valor de intensidad r_j y n_j

es el número de píxeles de la imagen que tienen el valor de intensidad j.

De esta manera, los píxeles s_k de la imagen procesada se obtienen al asignar a cada uno de los píxeles de la imagen de entrada, denotados por r_k, su respectivo valor obtenido por la transformación $T(r_k)$.

Reducción del ruido

Es muy común encontrar ruido aditivo en las imágenes médicas debido a su naturaleza y la forma en que son obtenidas. La disminución de este ruido puede resultar en una mejora en el rendimiento de clasificación, pues se trabaja con datos de entrada más limpios y de mejor calidad. Para reducir el ruido de una imagen es muy común utilizar filtros pasa-bajas, los cuales se encargan de suavizar la imagen reduciendo de esta manera el ruido presente en los píxeles (Gonzalez and Woods, 2008).

Una alternativa para la reducción de ruido son los filtros promediantes, los cuales son un tipo de filtros espaciales suavizantes que están conformados por una vecindad, la cual es un pequeño rectángulo de $m \times n$, donde normalmente m y n son iguales e impares. Los coeficientes del filtro llevan a cabo una operación definida que se realiza a los píxeles de la imagen abarcados por la vecindad. Dicha operación crea un nuevo valor para el píxel que se encuentra en el centro de la vecindad de la imagen filtrada. Este mismo procedimiento se debe seguir para obtener todos los nuevos valores de la imagen filtrada, a partir de un recorrido que se realiza a cada píxel de la imagen de entrada. La salida de estos filtros consiste en promediar los píxeles contenidos en la vecindad y, dependiendo de los coeficientes de la máscara del filtro, esta puede ser un promedio ponderado para ciertos píxeles dentro de la vecindad.

Otra opción son los filtros de orden estadístico. Al igual que los filtros promediantes, estos son un tipo de filtros espaciales, pero su principal diferencia se encuentra en el hecho de que estos ordenan los píxeles abarcados por la vecindad y reemplazan el valor del píxel central en la vecindad por el valor obtenido de aplicar una función estadística a los datos ordenados. Uno de los más conocidos es el filtro de la mediana el cual da buenos resultados al reducir el ruido de sal y pimienta.

Construcción de imagen a pseudocolor

Esta técnica es propuesta como parte del preprocesamiento utilizado en el trabajo de Heidari et al. (2020), la cual consiste en construir una imagen de tres canales RGB a partir de la integración de información diferente de una misma imagen en escala de grises, para cada uno de los canales:

- Canal rojo. Está conformado por la imagen en escala de grises ecualizada. Esto se hace debido a que las imágenes XR pueden presentar variaciones en el brillo debido a diversos factores externos, tales como la cantidad de rayos X utilizados para la obtención de

la imagen o el tamaño y volumen corporal del paciente. Mediante la ecualización del histograma se busca normalizar las imágenes y mejorar los patrones y características del tejido pulmonar asociadas a la enfermedad de COVID-19.

- Canal verde. Contiene la imagen a escala de grises original. Esta imagen se conserva original, con la finalidad de evitar la pérdida o modificación de la información relevante que se pueda presentar a causa de los filtros utilizados para la eliminación de ruido y la ecualización.

- Canal azul. Contiene la imagen después de aplicar la eliminación de ruido. Para esto se utiliza un filtro pasa bajas bilateral con *kernel* Gaussiano. Éste es un filtro no lineal similar al filtro promediante, el cual analiza los valores de intensidad de forma local y reemplaza la intensidad de cada píxel con el promedio de los píxeles de la vecindad. Sin embargo, tiene la característica de que los pesos del filtro son calculados por medio de una función Gaussiana. Este método se realiza con la finalidad de eliminar o disminuir el ruido aditivo común de las imágenes XR y como consecuencia obtener una imagen de mejor calidad.

La unión de los tres canales arriba descritos forman una imagen de *pseudocolor* con la cual se puede alimentar a un modelo de CNN. El objetivo de esta imagen es que la red tenga una mayor variedad de información para realizar una mejor distinción entre las clases.

2.3. Aprendizaje profundo

Uno de los modelos de aprendizaje computacional actuales más empleados para la clasificación de COVID-19 en el ámbito de la informática médica es el aprendizaje profundo. En esta sección se presentan algunos conceptos básicos de las Redes Neuronales Artificiales (ANN, por sus siglas en inglés) así como algunas arquitecturas de DNNs aplicadas a clasificación de COVID-19.

El desarrollo de modelos matemáticos que se inspiran en sistemas biológicos han sido un factor clave que ha impulsado el surgimiento de nuevas técnicas computacionales. Aquellos que se basan en la organización del sistema neuronal del cerebro son conocidas como *Redes Neuronales Artificiales*. El objetivo de estas redes es el aprendizaje a través de la experiencia y la obtención de conocimiento a partir de un conjunto de datos. Estos modelos emulan la conexión entre neuronas biológicas del cerebro, los cuales pueden ser considerados como una prueba de un sistema de procesamiento paralelo tolerante a fallas. Para modelar estas conexiones se utiliza un conjunto de unidades de cómputo conocidas como *neuronas artificiales*, las cuales se interconectan con otros elementos del mismo tipo y se distribuyen en estructuras conocidas como *capas*. De esta manera, la información viaja desde la entrada de la ANN hasta la salida, pasando por las diferentes capas que la conforman, las cuales se

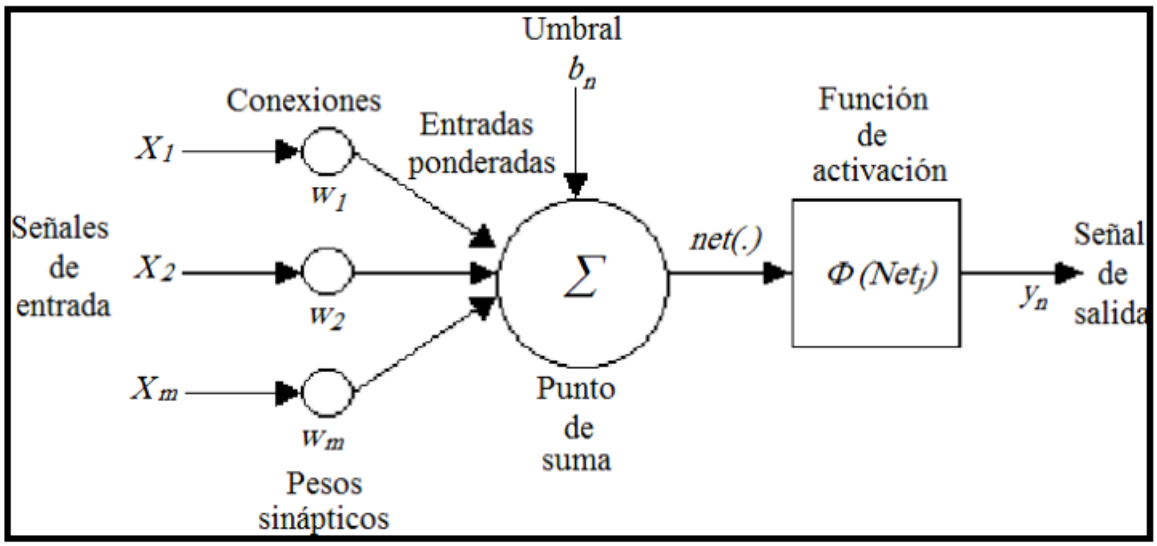


Figura 2.3: Esquema de una neurona artificial. Adaptado de (Haykin, 2007)

someten a operaciones matemáticas necesarias para obtener los valores de salida (Haykin, 2007).

Una neurona artificial es un modelo inspirado en una neurona biológica del cerebro humano, la cual está constituida por un conjunto de entradas que pueden considerarse como *enlaces sinápticos*, un punto de suma que produce un *potencial de activación*, una *función de activación* y una señal de salida, tal como se muestra en la Figura 2.3.

Cada uno de los componentes de la neurona artificial contribuye al funcionamiento de la misma y el de la red en general. A continuación se describen dichos elementos y las funciones de cada uno.

- Las señales de entrada reciben información que puede provenir de la entrada de la red o de otras neuronas artificiales. Estas señales viajan a través de los enlaces de conexión de la red, los cuales se encuentran parametrizados por sus pesos sinápticos respectivos.

- El punto de suma se encarga de procesar y unir las señales ponderadas recibidas en cada una de las entradas de la neurona.

- La función de activación se encarga de transformar el potencial de activación obtenido por el punto de suma en la salida de la neurona. Algunas funciones de activación básicas son: la función de escalón o umbral, la función lineal, la función sigmoidal, entre otras.

- Finalmente, la señal de salida se puede transferir a los enlaces de conexión ponderados de las entradas de otras neuronas artificiales.

El funcionamiento de una ANN se puede dividir en dos fases: la fase de aprendizaje y la fase de operación. La fase de aprendizaje está definida por un esquema de aprendizaje, el cual determina el tipo de problemas que la red es capaz de resolver (Viñuela and Galván, 2004). El sistema de aprendizaje de las ANNs se basa en ejemplos y estos ejemplos están directamente ligados a la capacidad de la red para resolver un problema. La relevancia que tienen los datos de ejemplo para la fase de entrenamiento está dada por las siguientes características:

- Deben ser significativos: el conjunto de datos para el aprendizaje debe contener elementos suficientes para que la red sea capaz de ajustar los pesos sinápticos de forma eficaz.

- Deben ser representativos: los ejemplos deben contener las características esenciales para discernir entre clases de manera correcta. Además, los datos de ejemplo deberán estar distribuidos con un relativo balance, pues de lo contrario, la red tenderá a especializarse en el subconjunto de datos con mayor número de elementos.

Por lo general, el aprendizaje de la red consiste en realizar el ajuste de los pesos sinápticos, partiendo de un conjunto de pesos sinápticos inicializados aleatoriamente o siguiendo una distribución específica y buscando el conjunto de pesos que mejor se adapte a la tarea en cuestión. Para lograr esto, se introducen de forma iterativa todos los ejemplos del conjunto de aprendizaje y mediante algoritmos basados en métodos numéricos, se busca minimizar la función de costo o error de la red; al mismo tiempo se ajustan los pesos sinápticos de la misma. El proceso es repetitivo hasta que se cumple cierto criterio de convergencia.

El esquema de aprendizaje determina las características y la forma en que la red va a aprender, siendo los más conocidos el aprendizaje supervisado y no supervisado (Viñuela and Galván, 2004).

- Aprendizaje supervisado. Los datos del conjunto de entrenamiento cuentan con dos elementos fundamentales. El primero es la información de las muestras en cuestión y el segundo se refiere a información relativa a la solución del problema para cada una de las muestras. Para el caso de clasificadores, el segundo elemento es la etiqueta de la clase a la cual pertenece cada una de las muestras de entrenamiento, por lo que el aprendizaje supervisado utiliza esta información para ajustar los pesos sinápticos.

- Aprendizaje no supervisado. En este esquema sólo se cuenta con la información de los ejemplos de entrenamiento y se carece de información que permite guiar el proceso de aprendizaje, por lo que la modificación de los pesos sinápticos se realiza únicamente a partir de información interna. A los modelos que utilizan este esquema también se le conoce como modelos autoorganizados y se busca que la red identifique rasgos significativos, regularidades o redundancias en los datos de entrada.

La fase de prueba o validación se realiza una vez que la red ha sido entrenada de manera satisfactoria, por lo que los pesos sinápticos y la estructura de la red se *congelan*, quedando una red fija que será utilizada para el procesamiento de datos. Es en esta fase donde se determina el rendimiento de la red para discernir o generalizar conceptos o características entre los datos de entrada. Para obtener un resultado real, el conjunto de datos utilizado en esta fase debe ser independiente al conjunto de entrenamiento y debe contener únicamente ejemplos que la red no conozca o que no se le hayan presentado en el proceso de aprendizaje.

Tipos de redes neuronales artificiales

Las ANNs se pueden categorizar de diferente manera de acuerdo con sus características como son: según el tipo de conexiones, según el grado de conexión y según el número de capas (Rivas-Asanza et al., 2018).

Por el tipo de conexiones, se pueden encontrar dos modelos: las redes neuronales recurrentes y las no recurrentes. En las no recurrentes, la información viaja de la entrada hacia la salida, sin ningún tipo de retroalimentación. En cambio, para una red recurrente, las conexiones entre las neuronas están retroalimentadas con neuronas de capas diferentes, de la misma capa e incluso con la misma neurona en cuestión.

Con respecto al grado de conexión, se encuentran las redes totalmente conectadas y parcialmente conectadas. Como su nombre lo indica, las redes parcialmente conectadas no tienen una conexión total con las neuronas de otra capa; mientras que en las redes totalmente conectadas cada neurona de una capa tiene conexión directa con todas las neuronas de la capa siguiente.

Las redes clasificadas según el número de capas se pueden dividir en: redes neuronales monocapas y multicapas. La primera corresponde al modelo de red más sencillo, ya que consta de una única capa que se encarga de proyectar la capa de entrada a la capa de salida. Este modelo suele utilizarse para eliminar distorsiones de señales. La segunda es muy similar a la anterior, con la diferencia de que para esta existe un conjunto de capas intermedias entre la capa de entrada y la de salida. A este conjunto de capas se le conoce como *capas ocultas*. Las neuronas pertenecientes a las capas ocultas pueden estar parcial o totalmente conectadas entre capas. Una red neuronal multicapa que posee una gran cantidad de capas ocultas se le conoce como DNN.

Considerando a las DNNs, se ha encontrado que las técnicas de aprendizaje profundo han logrado una gran aceptación en el ámbito del aprendizaje computacional. Esto se debe en gran parte a que, los modelos computacionales que las componen les permiten tener una mayor capacidad para aprender las representaciones de datos en múltiples niveles de abstracción (LeCun et al., 2015).

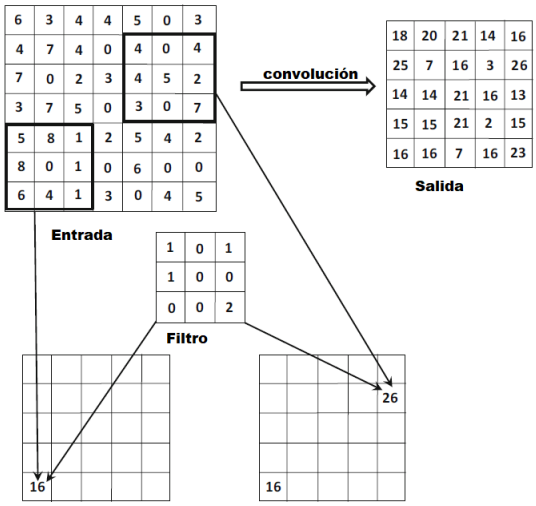


Figura 2.4: Ejemplo de una operación de convolución con un filtro de 3×3 sobre una cuadrícula de 7×7. Adaptado de (Aggarwal, 2018)

Redes Neuronales Convolucionales

Las CNNs son un tipo de DNN diseñadas para operar con entradas de dos dimensiones, cuyas regiones locales contienen dependencias espaciales importantes. Un ejemplo de este tipo de datos son las imágenes bidimensionales, las cuales tienen dependencias espaciales denotadas por la relación de las intensidades, formas y una gran variedad de características. Al trabajar con una dimensión extra, una CNN puede capturar los diferentes canales de una imagen, por ejemplo, si es una imagen a color la entrada es un volumen de tres dimensiones. Regularmente, este tipo de ANNs se centra en el procesamiento de datos de tipo imágenes aunque también pueden trabajar con otros tipos de datos como texto, series temporales y secuencias (Aggarwal, 2018).

La principal característica de las CNNs es que su funcionamiento se basa en el uso de una operación conocida como *convolución*. La convolución es una operación de producto escalar, la cual se realiza entre un conjunto de pesos sinápticos agrupados en estructuras con forma de cuadrículas y regiones de las entradas obtenidas de diferentes localidades espaciales y con estructuras del mismo tipo (ver Figura 2.4). Las CNNs toman su nombre de esta operación en específico, por lo que cualquier CNN implementa la operación de convolución en al menos una de sus capas, aunque en la práctica, la mayoría de modelos de CNNs implementan esta operación en múltiples capas.

Una CNN está conformada por tres tipos básicos de capas: convolución, submuestreo y la Unidad Lineal Rectificada (ReLU, por su acrónimo en inglés), las cuales se encuentran intercaladas y se pueden concatenar un número finito de veces. Al final de la red se suele agregar capas completamente conectadas junto con una capa de salida. Las diferentes capas

de una CNN se describen a continuación (Aggarwal, 2018).

Capa de convolución

La capa de convolución, como su nombre indica, es la encargada de aplicar la operación de convolución sobre las estructuras que recibe de entrada. Estas entradas siguen una organización tridimensional. Las dos primeras dimensiones definen las relaciones espaciales y la tercera dimensión las propiedades independientes extraídas de diferentes regiones locales de la cuadrícula de datos (características).

La operación de *convolución* consiste en superponer el filtro en cada una de las posiciones posibles de la imagen y por medio de un producto escalar ponderado obtener el píxel correspondiente de la cuadrícula de salida. Estos filtros se organizan en estructuras tridimensionales cuadradas de tamaño pequeño y normalmente impares cuyas dimensiones suelen ser de tamaño 1×1 o mayores, aunque lo más común es que sean de tamaño 3×3 o 5×5.

Debido a la operación de convolución y los filtros empleados, se forman las capas convolucionales, las cuales también cuentan con un parámetro de profundidad que está dado en relación con el número de filtros utilizados (con parámetros independientes). Cada una de estas capas se encarga de obtener las relaciones espaciales más importantes de la cuadrícula procesada, por lo que cada una de estas relaciones se genera como una nueva cuadrícula y se agrupa con el resto. A esto se le conoce como el *mapa de características*. Utilizar un mayor número de filtros para una capa convolucional implica obtener una mayor cantidad de mapas de características, es decir, una mayor profundidad para la capa oculta. El propósito de cada uno de los filtros es identificar un tipo de patrón espacial particular en determinada región de la imagen.

En ocasiones, se desea reducir la dimensión espacial de la cuadrícula de entrada de la capa convolucional con el objetivo de concentrar la información de las características y aumentar el campo receptivo de cada una de ellas; esto significa que, los filtros convolucionales de capas donde la cuadrícula ha sido reducida tienen la capacidad de obtener características que corresponden a una mayor región espacial en la cuadrícula antes de reducirla. Esto permite capturar características más complejas que se encuentran en regiones espaciales más grandes en la imagen.

Para realizar estas reducciones de una forma controlada, se utiliza la técnica de pasos o zancadas (*strides*). Esta técnica consiste en realizar la convolución sólo en ciertas ubicaciones de la cuadrícula, separadas en relación al valor del paso, lo que da como resultado una reducción de las dimensiones espaciales de la cuadrícula en un factor aproximado al paso utilizado. Regularmente, se utiliza un paso de 2 lo que provoca que cada una de las dimensiones espaciales de la imagen de entrada se reduzcan a la mitad. Regularmente, las CNNs suelen utilizar un paso de 2 en las capas que implementan esta técnica y, aunque no es muy común utilizar pasos mayores a 2, pueden ser útiles en entornos limitados de memoria o para reducir la dimensión espacial cuando esta es innecesariamente grande. Normalmente, las capas convolucionales se encuentran intercaladas con capas de submuestreo y capas de

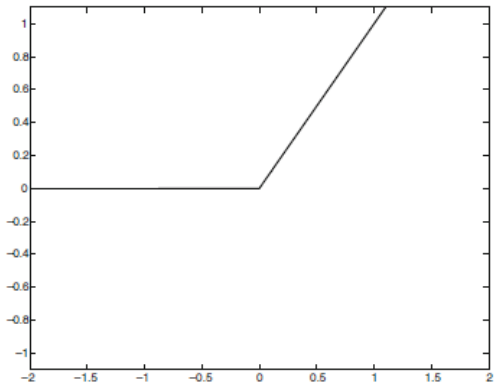


Figura 2.5: Gráfica de la función ReLU. Fuente: (Aggarwal, 2018)

activación.

Capas ReLU

La ReLU es una función de activación que se encarga de conservar únicamente los valores positivos sin modificar la entrada, mientras que los valores negativos son mapeados a cero. Esta función se define en la Ecuación 2.2.

$$F(v) = max(v, 0) \tag{2.2}$$

La función ReLU también es conocida como función de rampa debido a la forma que presenta en su gráfica (ver Figura 2.5). La capa ReLU en una CNN combina la función de activación con una transformación lineal que contiene una matriz de pesos para mapear los datos de entrada antes de pasarlos a la siguiente capa. De esta manera, se eliminan los valores negativos, los cuales carecen de importancia en operaciones con imágenes. El uso de esta función de activación se popularizó gracias a que demostró dar mejores resultados (en tiempos más cortos) en comparación con otras funciones de activación.

Capas de submuestreo

La operación de submuestreo (*pooling*) utiliza vecindades de tamaño pequeño similares a los filtros convolucionales, con la diferencia de que sus vecindades solo delimitan la región de la operación y no contienen ningún peso. Para cada región, la operación devuelve el valor máximo, mínimo o promedio contenido en la región, según el tipo de submuestreo que se utilice. El centro de cada una de las regiones de operación se encuentra separada por un paso (*stride*) mayor o igual a 1 (ver Figura 2.6). Esta operación se realiza de forma independiente para cada mapa de características generando un nuevo mapa con la misma profundidad. Dada la naturaleza de esta operación, es muy común utilizar regiones de 2×2 y un paso de 2 para generar una propiedad conocida como *invariancia a la traslación*, lo cual significa que mover ligeramente la imagen no produce cambios muy significativos en el

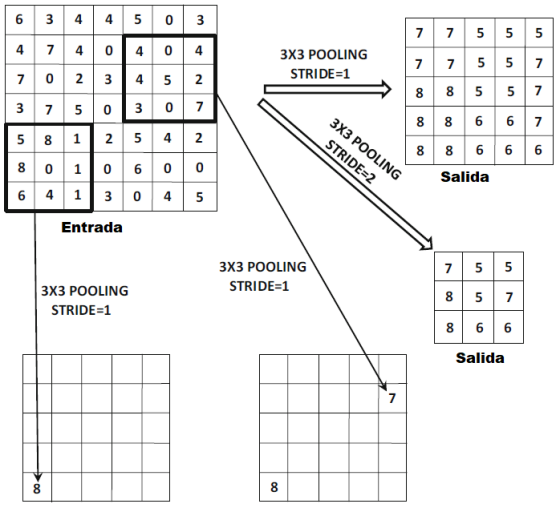


Figura 2.6: Ejemplo de un submuestreo máximo (*max-pooling*) a una cuadrícula de 7×7 utilizando vecindades de 3×3, aplicando paso 1 y paso 2. Adaptado de (Aggarwal, 2018)

mapa de salida. Esto permite detectar o clasificar objetos similares aunque se encuentren en diferentes regiones de la imagen.

Capas completamente conectadas

Las últimas capas de una CNN corresponden a capas completamente conectadas, las cuales funcionan de la misma manera que una red *feed-forward* tradicional y finalizan con una capa de neuronas de salida. Estas capas son las encargadas de realizar la tarea de clasificación, por lo que es necesario implementar al menos una capa completamente conectada. Dependiendo de la tarea o para mejorar la eficiencia de la red es posible utilizar más de una de estas capas. De manera similar que en las ANNs, abusar del número de capas completamente conectadas o del número de neuronas en cada una de ellas puede resultar perjudicial, ya que al tratarse de capas completamente conectadas, cada neurona de cada capa se conecta con todas las neuronas de la capa siguiente, lo que implica un gran número de parámetros. El cálculo de estos parámetros implica un costo computacional significativo respecto al uso de la memoria, por lo que ajustar el número de capas y unidades en cada una deben ser considerados como hiperparámetros relevantes al momento de diseñar la red. La capa de salida de una CNN se diseña de manera específica según la aplicación de la red o el problema a resolver. Por ejemplo, para el caso de la clasificación de imágenes es común que el número de neuronas en la capa de salida corresponda con el número de clases a identificar.

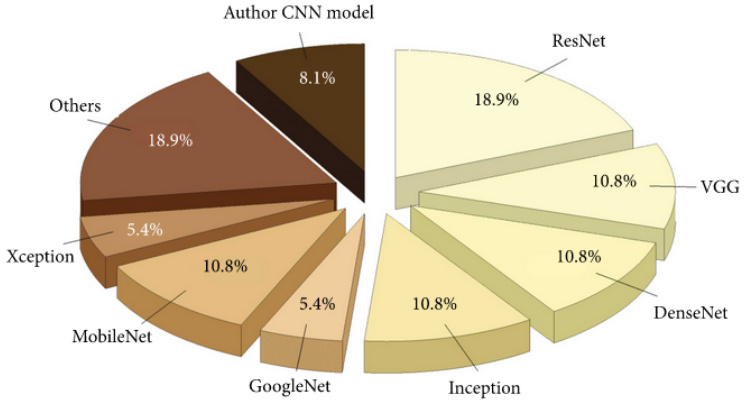


Figura 2.7: Estadística de uso de las arquitecturas de CNN para clasificación de COVID-19. Fuente: (Ghaderzadeh and Asadi, 2021)

2.3.1. Redes neuronales profundas en clasificación de COVID-19

Los resultados obtenidos de clasificación de COVID-19 mediante el uso de CNNs suelen ser satisfactorios. Sin embargo, existen diferentes arquitecturas de estas redes, cada una con un grupo de versiones o modelos establecidos, los cuales van directamente ligados con su eficacia. El rendimiento de clasificación de estos modelos puede variar según el problema para el cual se utilizan pues, según su arquitectura, algunos modelos se adaptan mejor a ciertas tareas.

En el trabajo de Ghaderzadeh and Asadi (2021) se realiza un análisis estadístico sobre el uso de diferentes técnicas utilizadas para la clasificación de COVID-19, considerando 160 artículos publicados desde el inicio de la pandemia hasta finales del año 2020. Como se puede observar en la Figura 2.7, los resultados muestran que los modelos de CNN más utilizados son aquellos que pertenecen a la familia de las Redes Residuales (ResNet, por su acrónimo en inglés), seguidos de otros modelos famosos tales como la familia VGG, DenseNet e Inception. La razón de esta distribución se debe a diversos factores, aunque, los resultados de este trabajo solamente reflejan la frecuencia de uso y la preferencia de los autores a dichos modelos.

En contraste, el trabajo de Lopez-Betancur et al. (2021) refleja un mejor análisis comparativo en función del rendimiento de diferentes modelos de CNN utilizados para la clasificación de COVID-19. En dicho trabajo se prueban un total de 32 modelos de CNNs utilizando el mismo conjunto de datos y los mismos hiperparámetros para el entrenamiento. Los resultados muestran que los modelos con un mejor comportamiento son modelos derivados de la familia ResNet, específicamente, los tres modelos con el mejor rendimiento son la *Wide_resnet101_2*, *Resnext101_32x8d* y *Resnext50_32x4d*. En el Cuadro 2.1, se muestran los resultados de rendimiento de los 6 mejores modelos de CNN obtenidos en Lopez-Betancur et al. (2021).

Modelo	Precisión	Sensibilidad	Especificidad
Wide_resnet101_2	97.75	97.75	96.76
Resnext101_32x8d	97.75	97.75	96.40
Resnext50_32x4d	97.75	97.75	96.07
Inception_V3	97.69	97.67	97.16
Mnasnet1_0	97.58	97.50	97.03
Wide_resnet50_2	97.50	97.50	96.16

Cuadro 2.1: Resultado de desempeño de los 6 mejores modelos de CNNs para clasificación de COVID-19. Fuente (Lopez-Betancur et al., 2021)

El alto rendimiento de los mejores modelos encontrados se deben a sus características y arquitectura que se adapta de buena manera al problema de clasificación de COVID-19. Entonces, apoyados en la literatura consultada se ha decidido estudiar la familia ResNet, por lo que a continuación, se describen las características de las arquitecturas ResNet y sus variantes.

Redes residuales

Las redes residuales (*Residual Network*) fueron construidas tomando como inspiración aquellas neuronas biológicas que tienen conexiones con otras neuronas que se encuentran en regiones apartadas del cerebro y que no necesariamente corresponden a capas contiguas, siendo capaces de saltarse varias capas intermedias para transmitir la información. En este mismo sentido, este principio se aplicó a las CNNs, principalmente para resolver el problema del desvanecimiento del gradiente (se presenta cuando este valor se aproxima a cero y por consecuencia el ajuste de pesos es mínimo o nulo). Para ésto, se introducen saltos entre bloques de 2 o 3 capas convolucionales, los cuales se encargan de pasar la entrada del bloque directamente a la salida funcionando como mapas de identidad. De esta forma, se garantiza que la información de entrada persista en la salida del bloque. Estos bloques son conocidos como bloques residuales y conforman la unidad básica de las redes residuales (Figura 2.8). Cada una de las capas convolucionales que conforman el bloque residual está acompañada por una normalización por lotes y una función de activación ReLU (He et al., 2016).

Los modelos ResNet manejan dos tipos de bloques residuales principalmente (ver Figura 2.9):

- Construcción: Los bloques de construcción (Fig. 2.9a) están conformados por una pila de 2 capas convolucionales con filtros de 3×3 y similares en dimensión.
- Cuellos de botella: Los bloques de cuello de botella (Fig. 2.9b) constan de una pila de tres capas convolucionales en cascada y con filtros de dimensiones 1×1, 3×3,

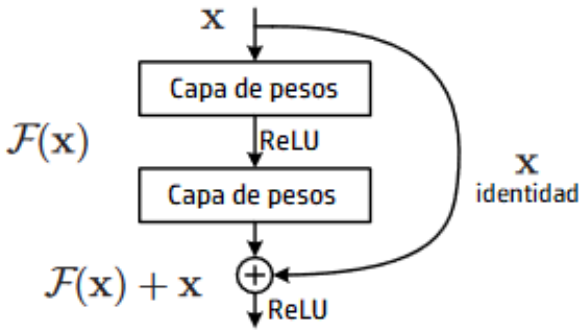


Figura 2.8: Bloque residual: unidad principal de las redes residuales. Adaptado de (He et al., 2016)

1×1. Su nombre se debe a que, cuando la entrada pasa por este bloque, este sufre una reducción en su dimensión antes de ser transformada por la capa convolucional de 3×3. Finalmente, se aplica la última capa convolucional de 1×1 para regresar a la misma dimensión que en la entrada del bloque.

Los saltos que se implementan con cada bloque residual se conocen como *conexiones de acceso directo*, debido a que saltan una o más capas. Para el caso de los modelos ResNet, estas conexiones realizan un mapeo de identidad, es decir, toman la información en la entrada del bloque y lo agregan de forma directa a la salida del mismo. Esta acción se denota por la Ecuación 2.3, pero para poder hacer esto es necesario que tanto la entrada como la salida tengan las mismas dimensiones.

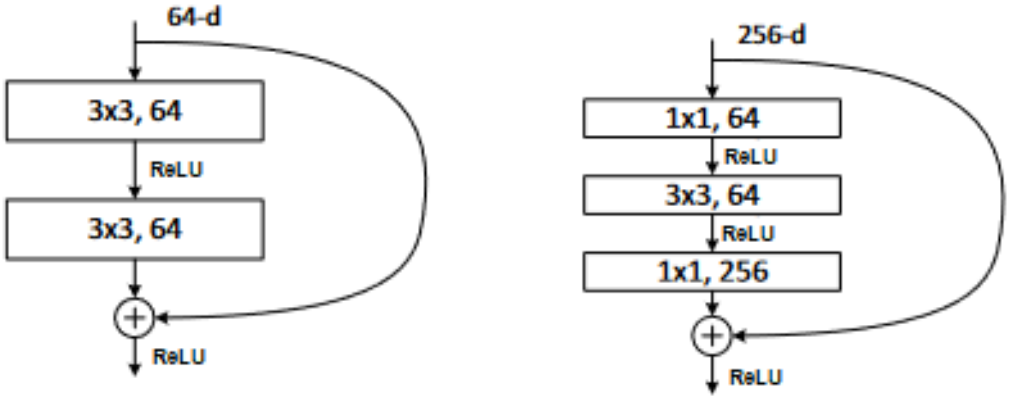
$$y = F(x) + x \tag{2.3}$$

donde y es la salida del bloque residual, $F(x)$ es la información transformada por las capas convolucionales que conforman el bloque residual, x es la información de entrada al bloque y "+" es una operación de concatenación.

Modelos ResNet

Utilizando los dos tipos de bloques residuales anteriormente mencionados como plantillas, He et al. (2016) propusieron los modelos de redes residuales que se muestran en el Cuadro 2.2.

En general, los modelos ResNet reciben como entrada imágenes con dimensión de 124×124 e inician aplicando una capa convolucional de 64 filtros de 7×7 con paso 2, seguida de una capa de submuestreo máximo con filtro de 3×3 y paso 2. A partir de este punto, la estructura de la red es diferente para cada modelo finalizando con una capa completamente conectada para la clasificación.



(a) (b)

Figura 2.9: a) Bloque residual de construcción básico. b) Bloque de cuello de botella. Fuente: (He et al., 2016)

Los modelos ResNet aquí descritos demostraron ser superiores en rendimiento a otras arquitecturas existentes al ganar la competencia de ImageNet en 2015. Uno de sus mayores impactos es que permitió por primera vez entrenar redes muy profundas (de más de 100 capas) afrontando de manera exitosa el problema del desvanecimiento del gradiente.

ResNeXt

Un problema recurrente en el diseño de nuevas arquitecturas de CNNs es el ajuste de los hiperparámetros, tales como el ancho de las capas convolucionales, el tamaño de los filtros convolucionales y de submuestreo, entre otros. Esto es significativo, especialmente cuando la red tiene un gran número de capas ocultas, ya que se pueden llegar a generar grandes cantidades de parámetros a estimar. Los modelos ResNet se construyen apilando bloques con la misma topología, favoreciendo a la reutilización de la misma topología repetidas veces para ayudar a reducir en gran medida el número de hiperparámetros, además de que puede llegar a reducir el sobreajuste de la red a un conjunto de datos específico.

La arquitectura ResNeXt se inspira en los bloques residuales de la ResNet para aprovechar la reducción de los hiperparámetros y propone modificaciones para mejorar el rendimiento de clasificación de la red sin aumentar la complejidad computacional ni el número de hiperparámetros (Xie et al., 2017).

Bloques residuales ResNeXt

Los modelos ResNeXt adoptan el principio de apilar bloques residuales de la misma topología e introducen dos reglas para la la construcción de estos:

Capa	Salida	18-capas	34-capas	50-capas	101-capas	152-capas
conv1	112×112	7×7, 64, paso 2				
conv2_x	56×56	3×3 MaxPool, paso 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$
	1×1	AveragePool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Cuadro 2.2: Plantilla para la implementación de modelos ResNet. Adaptado de (He et al., 2016)

- Si se producen mapas espaciales del mismo tamaño, los bloques comparten los mismos hiperparámetros (ancho y tamaño de filtro).

- Cada vez que se reduce la muestra del mapa espacial por un factor de dos, el ancho de los bloques se multiplica por un factor de dos.

La primera regla asegura que, tanto la complejidad computacional de la arquitectura como el número de hiperparámetros no sufran grandes variaciones; mientras que la segunda regla asegura que la complejidad computacional, en términos de Operaciones de Punto Flotante (FLOPs, pos sus siglas en inglés), sea similar para cada uno de los bloques de la red.

Estos nuevos bloques también se inspiran en los módulos de la CNN conocida como *Inception* presentados en Szegedy et al. (2016), en los cuales, por medio de capas convolucionales de 1×1, las entradas se dividen en incrustaciones (fragmentos de la entrada original de menor dimensión), las cuales sufren una transformación por medio de capas convolucionales especializadas con filtros de tamaño variado. Finalmente, cada una de las incrustaciones se fusionan mediante una concatenación para obtener una sola salida. A este comportamiento se le conoce como *división-transformación-fusión* y se ha demostrado que, una buena combinación puede llegar a compararse con representaciones de capas grandes y densas, pero con una complejidad computacional menor. Sin embargo, debido al uso de capas convolucionales especializadas, el número de los hiperparámetros suele ser sumamente grande, además de que pueden surgir problemas para adaptar estas arquitecturas a tareas distintas para las que fueron creadas.

Los nuevos bloques ResNeXt aprovechan la estrategia de división-transformación-fusión y la combinan con las topologías apiladas de los bloques ResNet. De esta forma, se logran

explotar las ventajas de los bloques Inception eliminando el problema de los hiperparámetros. Estos bloques funcionan de la siguiente manera:

- **División**: Por un lado, la entrada del bloque se divide en una *cardinalidad* de incrustaciones de baja dimensión mediante capas convolucionales de 1×1. Por otro lado, la dimensión d de cada incrustación está dada por la dimensión original de la capa convolucional dividida entre la cardinalidad del bloque.

- **Transformación**: Cada una de las incrustaciones es sometida a una transformación, la cual se establece como un bloque de cuello de botella donde la primera capa de 1×1 se encarga de generar la incrustación y la última capa se encarga de regresar a la dimensión de entrada.

- **Agregación**: Todas las incrustaciones se agregan para obtener una única salida. Este proceso puede verse como el producto interno modificado de una neurona artificial simple, donde el número de entradas está denotado por la cardinalidad, cada una de las incrustaciones representa una entrada a la neurona y la suma ponderada es reemplazada por la transformación de la incrustación. De esta manera, la operación de la neurona puede ser reformulada como una combinación de división, transformación y agregación, obteniendo la ecuación 2.4.

$$F(x) = \sum_{i=1}^{C} \tau_i(x) \tag{2.4}$$

  donde C es la cardinalidad, i denota cada una de las incrustaciones, y $\tau_i(x)$ representa la transformación individual de cada entrada.

  Para este bloque, la ecuación 2.4 funciona como una función residual, al agregar la conexión de mapeo de identidad, la función residual queda de la siguiente manera (Ecuación 2.5):

$$y = x + \sum_{i=1}^{C} \tau_i(x) \tag{2.5}$$

  donde x es la entrada del bloque y y es la salida.

En el Cuadro 2.3 se observa la arquitectura ResNeXt-50 comparada con la arquitectura ResNet-50 y se puede observar que, el número de parámetros y de FLOPs se mantienen cercanos.

Los modelos ResNeXt han demostrado que la cardinalidad es una dimensión que tiene una importancia similar al ancho y profundidad de las redes, ya que aumentar la cardinalidad resulta una forma más efectiva de ganar exactitud, incluso más que aumentando la

Etapa	Salida	ResNet-50	ResNeXt-50 (32×4d)
conv1	112×112	7×7, 64, paso 2	7×7, 64, paso 2
conv2	56×56	3×3 MaxPool, paso 2	3×3 MaxPool, paso 2
		$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128, C=32 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3	28×28	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256, C=32 \\ 1\times1, 512 \end{bmatrix}\times4$
conv4	14×14	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512, C=32 \\ 1\times1, 1024 \end{bmatrix}\times6$
conv5	7×7	$\begin{bmatrix} 1\times1, 1024 \\ 3\times3, 1024 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512, C=32 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	GlobalAveragePooling, 1000-d fc, softmax	GlobalAveragePooling, 1000-d fc, softmax
# parámetros		25.5×10^6	25.0×10^6
FLOPs		4.1×10^9	4.2×10^9

Cuadro 2.3: Arquitectura del modelo ResNet-50 en comparación con la arquitectura del modelo ResNeXt-50. Adaptado de (Xie et al., 2017)

profundidad o el ancho, especialmente cuando estas dos dimensiones comienzan a generar problemas en el rendimiento de la red.

Wide_resnet

Si bien se ha demostrado que las redes residuales presentan buenos resultados en el rendimiento de clasificación gracias a los bloques residuales, también se ha encontrado que éstos presentan algunas falencias. La principal falencia de las redes residuales se presenta en la propagación hacia adelante, donde los mapas de características calculados por las capas anteriores pueden eliminarse o desvanecerse a causa de multiplicaciones repetidas con las matrices de pesos. Este fenómeno genera una mayor dificultad para la red de obtener direcciones de gradientes significativas. A pesar de que los mapas de identidad en los bloques residuales solucionan esto, se tiene que la salida en bloques con este problema sigue siendo poco significativa. Esto puede provocar que varios de los bloques residuales aporten muy poca información relevante haciendo que su contribución al aprendizaje sea mínima, lo que implica un gasto de tiempo de aprendizaje desaprovechado. A este problema se le conoce como la disminución de la reutilización de características.

Las redes residuales amplias (Wide_resnet, por su acrónimo en inglés) trabajan sobre los bloques residuales e introducen dos nuevos factores: El factor de profundización l, que denota el número de convoluciones en un bloque residual, y el factor de ampliación k, el cual

Etapa	Salida	Tipo de bloque = $B(3,3)$
conv1	32×32	$[3 \times 3, 16]$
conv2	32×32	$\begin{bmatrix} 3 \times 3, 16 \times k \\ 3 \times 3, 16 \times k \end{bmatrix} \times$N
conv3	16×16	$\begin{bmatrix} 3 \times 3, 32 \times k \\ 3 \times 3, 32 \times k \end{bmatrix} \times$N
conv4	8×8	$\begin{bmatrix} 3 \times 3, 64 \times k \\ 3 \times 3, 64 \times k \end{bmatrix} \times$N
avg-pool	1×1	$[8 \times 8]$

Cuadro 2.4: Plantilla para la implementación de un modelo Wide_resnet. Adaptado de (Zagoruyko and Komodakis, 2016)

multiplica el número de características en las capas convolucionales. Además, se define como d al total de bloques residuales de la red y n al total de capas convolucionales (Zagoruyko and Komodakis, 2016).

Bloques residuales amplios

La principal potencia de los modelos ResNet está dada por sus bloques residuales. Por este motivo, las redes residuales amplias buscan aumentar el poder de estos bloques de tres maneras: primera, agregando un mayor número de capas convolucionales por bloque; segunda, ampliando las capas convolucionales al incrementar el número de planos de características; y tercera, aumentando los tamaños de los filtros en las capas convolucionales.

Para representar estos nuevos bloques residuales se utiliza la notación $B(M)$, donde M es una lista con los tamaños de los filtros de cada capa convolucional. De esta manera $B(3,3)$ representa un bloque residual con dos capas convolucionales de 3×3, correspondiente al bloque de construcción básico de la ResNet y $B(1,3,1)$ representa un bloque convolucional de cuello de botella.

De esta manera, las plantillas para los modelos Wide_resnet se representan según se observa en el Cuadro 2.4, donde $B(3,3)$ representa el tipo de bloque residual que será usado, k es el factor de ampliación que multiplica la dimensión de las capas convolucionales y N es la cantidad de bloques residuales de cada tipo.

Para afrontar el problema de disminución de la reutilización de características, los bloques residuales amplios implementan capas de abandono (*dropout*) entre las convoluciones, la cual también ayuda a evitar el sobreajuste de la red.

Estas redes han demostrado ser considerablemente mejores que otros modelos existentes para una gran variedad de tareas (Song et al., 2021; Tan et al., 2021; Yang et al., 2019; Panda et al., 2019), demostrando que, con un factor de ampliación $k \geq 2$, puede llegar a superar a redes más profundas (Lopez-Betancur et al., 2021; Zagoruyko and Komodakis, 2016; Zhong

et al., 2020).

MFL_ Net

Como se muestra al inicio de esta sección, las redes residuales demuestran una gran efectividad al momento de trabajar con los problemas de clasificación de COVID-19, ésto gracias a que los bloques residuales permiten la creación de redes con una gran profundidad. Si bien esto es una gran ventaja, también conlleva algunos puntos negativos. Los modelos residuales con mejor rendimiento son aquellos con una profundidad de 50 o superior. Esta gran cantidad de bloques residuales implica un alto número de parámetros en la red, lo que se traduce en un mayor tiempo de entrenamiento en comparación con modelos más ligeros.

Una alternativa a las redes residuales es la red *MFL_ Net*, la cual es un modelo CNN de poca profundidad y significativamente liviano, implementado para problemas de clasificación de COVID-19 en imágenes de CT. Al ser un modelo pequeño, se ve beneficiada con un número de parámetros considerablemente bajo y un tiempo de entrenamiento reducido. Además, demuestra ser un modelo eficiente gracias a la implementación de aprendizaje multiescala (Joshi and Nayak, 2022). Los modelos CNN que implementan una estructura apilada lineal para la conexión de sus capas convolucionales normalmente se enfocan en la extracción de características en una escala homogénea. Sin embargo, la extracción de características a diferentes escalas ha demostrado una gran efectividad en las tareas de clasificación (Li et al., 2019).

Bloques de funciones de aprendizaje multiescala

El buen rendimiento de las redes MFL radica en los bloques de Funciones de Aprendizaje Multiescala (MFL, por sus siglas en inglés), los cuales funcionan como un módulo de extracción de características a diferentes escalas. Estos bloques tiene como objetivo el aprendizaje de características a diferentes niveles del campo receptivo. Para esto implementan la estructura presentada en la Figura 2.10.

La unidad básica de los bloques MFL son las estructuras denominadas *Mini Block*, las cuales consisten en un secuencia de una capa convolucional, una capa de normalización y la capa de activación. Los bloques MFL se encargan de extraer las características en dos diferentes escalas de manera simultánea, estas son:

- Escala pequeña: se obtiene utilizando filtros de tamaño 3×3 y se encargan de la extracción de características detalladas. Estos filtros son los encargados de obtener las características de la enfermedad de COVID-19, tales como los parches de opacidades de vidrio esmerilado y las opacidades lineales.

- Escala grande: está representada por filtros de tamaño 5×5 y 7×7, las cuales se encargan de obtener las características generales como la forma de los pulmones.

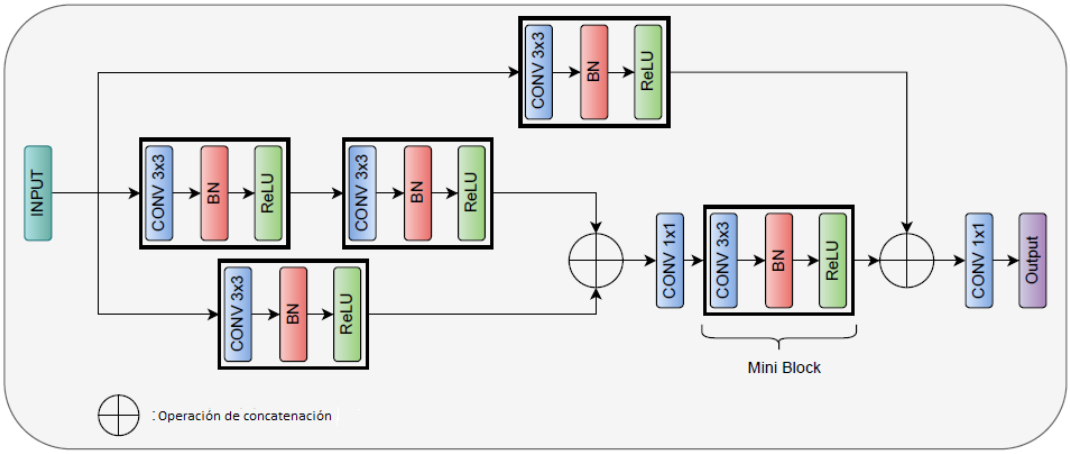


Figura 2.10: Estructura del bloque MFL. Fuente: (Joshi and Nayak, 2022)

Un problema al utilizar filtros de múltiples tamaños es que el número de parámetros se incrementa de manera significativa y proporcional según se aumenta el tamaño de los filtros. Para afrontar ésto, los bloques MFL proponen simular las capas convolucionales de escala grande mediante la combinación secuencial de varios *Mini Blocks*. Específicamente, se utilizan dos *Mini Blocks* de manera secuencial para simular una capa convolucional de 5×5 y tres en el caso de un capa convolucional de 7×7; la arquitectura resultante es la mostrada en la Figura 2.10. Esta implementación optimiza el funcionamiento de la red al reducir de manera significativa el número de parámetros.

Sin duda, las DNNs han tenido un gran impacto y aceptación en cuanto a clasificación de COVID-19 se refiere. Las CNNs se han convertido en uno de los modelos de aprendizaje profundo más utilizados gracias a los buenos resultados que presenta al trabajar tanto con imágenes XR como con CT.

Módulo de atención de bloque convolucional

El Módulo de Atención de Bloque Convolucional (CBAM, por sus siglas en inglés) es un módulo liviano que puede integrarse a cualquier modelo de CNN para obtener una mejora en la capacidad de interpretabilidad de las arquitecturas. El mapa de atención obtenido es multiplicado por un mapa de características de entrada, lo cual permite un refinamiento de características adaptativo (Woo et al., 2018). Este módulo está conformado por dos submódulos de atención secuenciales, uno para cada tipo de información, los cuales son: el submódulo de atención de canal y el submódulo de atención espacial (ver Figura 2.11).

El submódulo de atención de canal considera cada canal de un mapa de características como un detector de características y utiliza la relación de características existente entre canales para generar un mapa de atención del canal. Para realizar el cálculo del mapa de

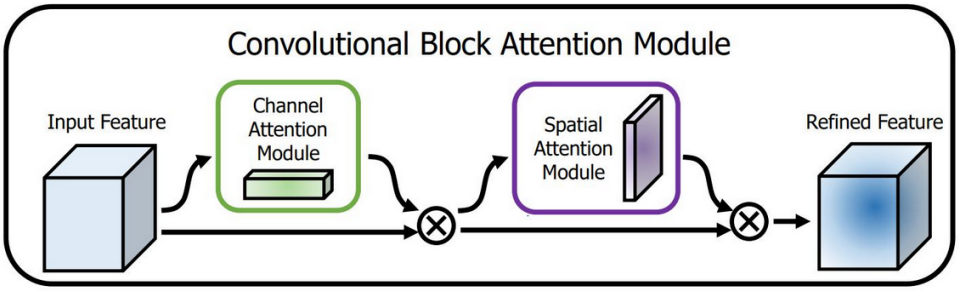


Figura 2.11: Estructura del módulo CBAM. Fuente: (Woo et al., 2018)

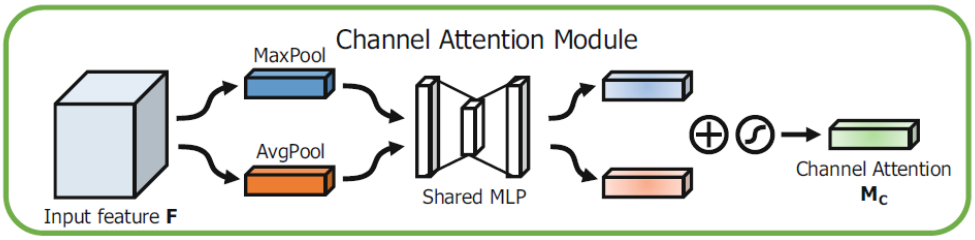


Figura 2.12: Estructura del submódulo de atención de canal. Fuente: (Woo et al., 2018)

atención del canal es necesario comprimir la dimensión espacial del mapa de características de entrada. Ésto se hace utilizando operaciones de submuestreo promedio y máximo para generar dos descriptores de contexto espacial diferentes, los cuales son enviados a una red perceptrón multicapa con una capa oculta para producir los mapas de atención del canal. Después de aplicar el perceptrón multicapa a cada descriptor, se utiliza una operación de concatenación para fusionar los vectores de características de salida y una operación de normalización con una función *sigmoide*, obteniendo un único mapa de atención de canal general. La estructura completa del submódulo de atención de canal se muestra en la Figura 2.12.

En el caso del submódulo de atención espacial, se utiliza la relación espacial entre características para generar un mapa de atención espacial. La atención espacial se centra en determinar las regiones más relevantes de la entrada, por lo que puede considerarse como un complemento de la atención de canal. Siguiendo la estructura mostrada en la Figura 2.13, el cálculo de la atención espacial se realiza por medio del submuestreo promedio y submuestreo máximo generando dos mapas bidimensionales (uno por cada tipo de submuestreo); posteriormente, se utiliza una operación de concatenación y convolución básica para generar el mapa de atención espacial 2D. Finalmente, mediante la función *sigmoide* se realiza la normalización del mapa de atención obtenido.

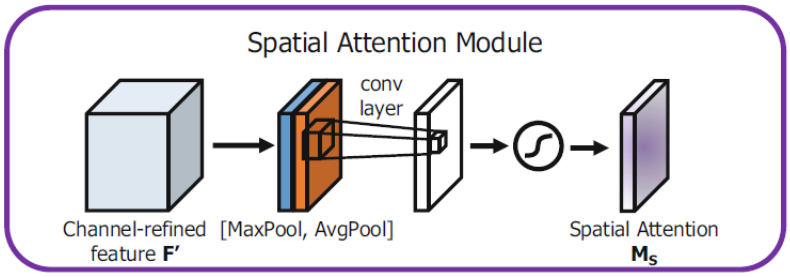


Figura 2.13: Estructura del submódulo de atención espacial. Fuente: (Woo et al., 2018)

2.4. Optimización de redes profundas mediante metaheurísticas

En esta sección se presentan algunos factores relacionados con la optimización de las DNNs, iniciando con una breve descripción de los hiperparámetros de las DNNs, seguido de algunos métodos de optimización y finalizando con algunos ejemplos sobre la implementación de la optimización y su rendimiento.

Cabe mencionar que, el término heurística, en el área de inteligencia artificial, hace alusión a los procedimientos que obtienen soluciones a problemas de manera inteligente por medio del uso de conocimiento y experiencia, lo que ha otorgado buenos resultados en cuanto a calidad y velocidad. Algunas heurísticas han resultado ser exitosas para resolver problemas específicos, por lo que se han expandido para aplicarlas como estrategias generales en la resolución de problemas. A dichas estrategias se les ha denominado como metaheurísticas (Melián et al., 2003). Dados los buenos resultados que estos métodos presentan para resolver problemas de búsqueda combinatoria, resultan una alternativa viable y competitiva para realizar la búsqueda de valores óptimos de los hiperparámetros de una red neuronal con la intención de maximizar su rendimiento.

2.4.1. Métodos de optimización de hiperparámetros para redes profundas

Los hiperparámetros de una ANN son aquellos parámetros que se encuentran relacionados tanto con la estructura de la red como con el algoritmo de entrenamiento de la misma, los cuales influyen de manera directa con el rendimiento de la red neuronal. Estos hiperparámetros se deben de establecer al momento de implementar la red neuronal (Probst et al., 2019). Sin embargo, estas redes deben adaptarse al problema de investigación y al tipo de datos con los que están trabajando, por lo que su estructura interna, así como su algoritmo de entrenamiento puede variar según el problema de investigación lo requiera. Por esta razón, se presentan problemas al momento de seleccionar aquellos hiperparámetros de la red que generen resultados óptimos (Sánchez-Caballero, 2020).

Entonces, el rendimiento de una ANN está ligado a sus hiperparámetros, por lo que, encontrar los valores óptimos de estas resulta complicado debido a que se requiere especificar cierto conjunto de valores. Por lo tanto, probar con todas las combinaciones posibles de hiperparámetros se traduce en un alto costo computacional (Aszemi and Dominic, 2019).

Este problema se acentúa aún más cuando la cantidad de hiperparámetros que puede tener una ANN es tan grande que resulta muy complicado obtener las valores óptimos de forma manual; tal es el caso de las DNNs. Algunos de los hiperparámetros que pueden optimizarse (Goel et al., 2021) son, por ejemplo, los siguientes:

- Tasa de aprendizaje: controla la velocidad del descenso del gradiente en el algoritmo de entrenamiento.
- Número de épocas: determina el número de veces que el algoritmo de aprendizaje actualizará los parámetros de la red.
- Impulso (momentum): controla la influencia de la actualización de los pesos anteriores en la actualización de los pesos actuales.
- Coeficiente de regularización: regula el problema del sobreajuste en la red.

Otro ejemplo de hiperparámetros a optimizar es el presentado en Serizawa and Fujita (2020), quienes utilizaron un modelo de CNN con dos capas de convolución y dos capas densas para realizar pruebas de clasificación de imágenes y reconocimiento de dígitos. En este caso, los autores decidieron optimizar los siguientes hiperparámetros de la red:

- Número de filtros para cada capa de convolución.
- El tamaño del kernel de cada capa de convolución.
- La función de activación en cada capa de convolución.
- La función de activación en cada capa densa.
- El número de neuronas en cada capa densa.
- El número total de muestras de entrenamiento presentes en un solo lote.

Estos son algunos de los hiperparámetros que se suelen optimizar para mejorar el rendimiento de las DNNs. Sin embargo, no existe un esquema general para ajustar hiperparámetros, por lo que los trabajos presentados anteriormente son casos particulares cuyo único objetivo es mejorar el rendimiento de las redes.

Como ya se mencionó en párrafos anteriores, determinar el valor de los hiperparámetros de una DNN es un aspecto importante y la forma común de hacerlo es mediante el uso de valores predeterminados para modelos, paquetes de software, búsqueda de malla o la selección

de forma manual por parte del usuario. Por esta razón, diversos trabajos de la literatura recomiendan obtenerlos mediante algún procedimiento o algoritmo de ajuste automático (Probst et al., 2019).

Métodos tradicionales

Una de las primeras técnicas de optimización implementadas fue sobre el valor de la LR durante el entrenamiento de una ANN, esperando obtener una convergencia más rápida y por tanto una reducción del error más eficiente. La importancia de la LR radica en que es un hiperparámetro utilizado en la mayoría de los algoritmos de aprendizaje, pues es en relación a ésta que se puede influir en el ajuste de los pesos. La LR tiene la característica de que, cuando su valor es muy pequeño, la red obtiene una mayor precisión a costa de un incremento en los tiempos de procesamiento del entrenamiento. En cambio, cuando su valor es demasiado grande, el error puede reducirse más rápido, pero al acercarse a los valores óptimos, la función puede atascarse en un mínimo local y es posible que nunca alcance una solución (Kumar and Hati, 2021). Establecer un valor fijo para la LR significa elegir entre uno de estos dos aspectos: velocidad o precisión. Es por esto que, se implementaron diferentes técnicas para optimizar el ajuste de los parámetros, tales como el descenso de gradiente adaptativo, el método Adam, entre otros. Incluso la selección de alguno de estos métodos para el entrenamiento es considerada como otro hiperparámetro llamado optimizador, por lo que el problema se traslada a decidir cuál de estos métodos funciona mejor en combinación con los otros hiperparámetros seleccionados.

Otra técnica es la *búsqueda de malla* o cuadrícula, la cual consiste en realizar una búsqueda de los valores de los hiperparámetros previamente definidos. Esta búsqueda se realiza en una cuadrícula N dimensional, donde N es el número de hiperparámetros y cada ubicación en la cuadrícula corresponde a una combinación de hiperparámetros. En otras palabras, se realiza una búsqueda exhaustiva tratando de probar todas las combinaciones posibles. Sin embargo, este método resulta ser ineficiente para las CNNs dado el alto número de hiperparámetros que se pueden llegar a necesitar para optimizar (Mahdaddi et al., 2021).

La optimización Bayesiana surge como una técnica de búsqueda probabilística basada en resultados previos. Para este tipo de optimización, se toma como función objetivo el rendimiento de la red en función de alguna medida de rendimiento con los valores de los hiperparámetros seleccionados (Shahriari et al., 2015). El objetivo de este método consiste en guiar la búsqueda en cada iteración hacia un espacio de mayor interés. Esto se logra seleccionando con una mayor probabilidad a los valores candidatos que en iteraciones anteriores demostraron proporcionar buenos resultados y despreciando a los que no. Con esto, se logra reducir el número de combinaciones posibles para los hiperparámetros. El problema de este enfoque está dado por su complejidad, además de que su alto espacio dimensional se puede traducir en un alto costo computacional (Mahdaddi et al., 2021).

Dada la gran cantidad de hiperparámetros que se pueden elegir para realizar la optimización y la cantidad de combinaciones de los valores que se pueden llegar a generar, se necesitan

de algoritmos especializados en resolver problemas de búsqueda combinatoria, también llamados algoritmos de optimización. Una alternativa para esto son las *metaheurísticas*, las cuales son algoritmos que emplean conocimiento disponible acerca de un problema y de las posibles técnicas o estrategias que se pueden utilizar para tratar de acercarse a una solución en un tiempo razonable (Melián et al., 2003). Algunos de los métodos más conocidos son los algoritmos evolutivos.

Algoritmos genéticos

Los GA son aquellos cuyo funcionamiento se basan en las leyes de la selección natural, de tal forma que tratan de imitar el mecanismo de evolución de los organismos siguiendo el mismo principio de la evolución. Sólo aquellos que mejor se adaptan son los que sobreviven (Iba, 2018). La representación de la información de estos algoritmos se basa en dos estructuras denominadas genotipos y fenotipos. La primera es una estructura que contiene la información o representación de una solución haciendo analogía al código genético que se encuentra en los cromosomas dentro de las células. La segunda expresa el comportamiento de la solución en el problema, simulando el surgimiento de comportamiento en los organismos para adaptarse a su entorno. El marco básico de los algoritmos evolutivos se compone de los siguientes elementos (Iba, 2018):

- Los individuos. Cada individuo es una posible solución al problema de optimización sobre el que se está trabajando.
- La población. Está conformada por un conjunto de individuos que cambia en cada generación.
- Generación. Es el conjunto de individuos que conforman la población en cada iteración.
- Genotipo. El código genético de cada individuo (datos o información).
- Aptitud. Es una calificación que determina qué tan bueno es un individuo para solucionar un problema.
- Función objetivo. Es una función definida por el problema de optimización que se está tratando y se utiliza para calcular la aptitud de los individuos.
- Reproducción. Es un operador el cual consiste en realizar la combinación genética entre individuos conocidos como *padres* para obtener un individuo conocido como *hijo*.
- Mutación. Es una técnica utilizada para evitar el estancamiento en óptimos locales mediante una ligera modificación del material genético de ciertos individuos.

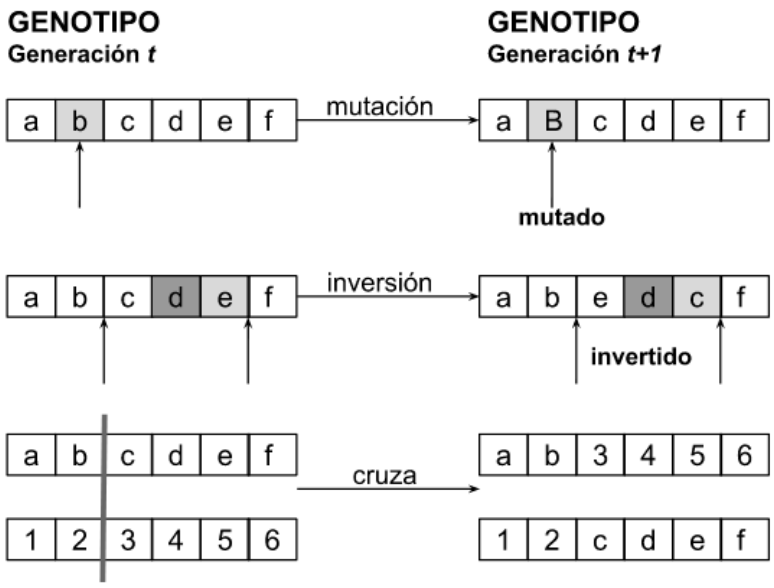


Figura 2.14: Operadores evolutivos. Adaptado de (Iba, 2018)

Los principales operadores de los algoritmos evolutivos son los ilustrados en la Figura 2.14, aunque entre tipos de algoritmos suelen tener variaciones en el orden de aplicación y en la forma de aplicarlos. Las operaciones de cruza, reproducción, o recombinación representan las principales formas de producir a los individuos que conformarán la nueva generación y es natural que se espere que estos mejoren su aptitud respecto a generaciones anteriores. Es por este motivo que, elegir a los padres es una tarea sumamente importante. Entre los métodos más comunes de selección de padres se encuentra el método de la ruleta y selección por torneo. El primero consiste en una selección aleatoria ponderada por la aptitud de cada individuo; de esta manera, aquellos con una mayor aptitud tendrán mayor probabilidad de ser seleccionados. En el segundo se selecciona de manera aleatoria un conjunto de posibles candidatos a la reproducción y se eligen a los dos más aptos. La estrategia de *elitismo* es un método de selección de individuos que se utiliza, en combinación a los métodos de reproducción, para constituir a la nueva generación. Este consiste en seleccionar a los mejores individuos de la generación actual y preservarlos íntegros para la siguiente generación.

En todo algoritmo de búsqueda existe el riesgo de estancamiento en óptimos locales. Para evitar ésto, se implementa el operador de mutación, el cual consiste en hacer una ligera modificación en el genotipo al momento en que se crea un hijo. Una operación simple que es considerada como mutación es el operador de inversión.

Evolución diferencial

La DE es un algoritmo que sigue el esquema de los algoritmos evolutivos, el cual realiza búsqueda multidimensional mediante el uso de mutación y la recombinación genética (Storn and Price, 1997). Al igual que otros algoritmos evolutivos como GA, la DE trabaja con una población de soluciones (individuos), donde cada individuo se encuentra representado como un vector de la misma dimensionalidad del problema y cada uno de los genes que lo compone se inicializa utilizando un generador de números aleatorios dentro de un determinado rango de búsqueda, tal y como se muestra en la Ecuación 2.6.

$$x_{i,t}^{G=1} = Random_t(LB_t, UB_t) \tag{2.6}$$

donde $G = 1$ es la generación inicial, t es el t-ésimo cromosoma del i-ésimo individuo, LB_t y UB_t son los límites inferior y superior del rango de la búsqueda y $Random_t(a, b)$ denota un generador aleatorio.

Una de las principales particularidades del algoritmo DE es la forma en que se realiza la reproducción. Principalmente, este algoritmo no utiliza mecanismos de selección de los padres, es decir, cada individuo tiene las mismas oportunidades de reproducirse y generar descendencia. Para esto, cada individuo de la población actual se convertirá en un padre y se le conoce como *padre principal* o *individuo objetivo*. Por cada padre principal se seleccionan, de manera aleatoria, a tres o más padres auxiliares que servirán para generar al *individuo mutado*. Para esto, se calcula la diferencia escalar entre dos de los padres auxiliares y se suma al tercero. A esta operación se le conoce como *mutación diferencial* y es de esta de donde éste algoritmo toma su nombre. La operación puede verse representada en la Ecuación 2.7.

$$\vec{v}_i^{G=1} = \vec{r}_1 + F \times (\vec{r}_2 - \vec{r}_3) \tag{2.7}$$

donde F es un número real positivo, $\vec{r}_1$, $\vec{r}_2$ y $\vec{r}_3$ son los padres auxiliares seleccionados de forma aleatoria y $\vec{v}_i^{G=1}$ es el individuo mutado.

El algoritmo DE utiliza un operador de cruce conocido como *recombinación discreta*, en la que el padre principal o individuo objetivo se cruza con el individuo mutado para generar un hijo. Este hijo se crea eligiendo genes de cualesquiera de los padres de forma aleatoria, con la única regla de que el hijo deberá heredar a lo mínimo un gen del padre mutado. La selección de los genes que conformaran al nuevo hijo se realiza siguiendo la Ecuación 2.8.

$$u_{i,t}^G = \begin{cases} v_{i,t}^G & \text{si } r(t) \leq C_r \text{ o } t = rn(i) \\ x_{i,t}^G & \text{si } r(t) > C_r \text{ y } t \neq rn(i) \end{cases} \tag{2.8}$$

donde cada gen $u_{i,t}^G$ del nuevo individuo $\vec{u}_i^G$ es obtenido de $x_{i,t}^G$ y $v_{i,t}^G$ según la *tasa de cruza* dada por C_r y un numero $r(t)$ generado de forma aleatoria, asegurando que al menos el gen en el índice $rn(i)$ del individuo $v_{i,t}^G$ sea heredado al nuevo individuo.

Finalmente, una vez que se ha generado el hijo, es momento de seleccionar a los individuos que conformaran la nueva generación. Para esto, el algoritmo DE realiza una competencia

entre cada padre $\vec{x}_i^G$ y su respectivo hijo $\vec{u}_i^G$, donde el menos apto de los dos es desechado, tal como se muestra en la Ecuación 2.9.

$$\vec{x}_i^{G+1} = \begin{cases} \vec{u}_i^G & \text{si } f(\vec{u}_i^G) \leq f(\vec{x}_i^G) \\ \vec{x}_i^G & \text{en cualquier otro caso} \end{cases} \tag{2.9}$$

donde $\vec{x}_i^{G+1}$ es el individuo que formara parte de la siguiente generación y $f(\cdot)$ denota una calificación obtenida mediante una función objetivo de minimización (o maximización, de acuerdo al problema a resolver).

Optimización de lobos grises

GWO es un algoritmo de manada inspirado en el comportamiento de los lobos grises y la estructura jerárquica por la cual se rige la manada (Mirjalili et al., 2014).

Los lobos grises son una especie de depredadores que tienden a vivir en grupos de entre 5 a 12 integrantes en promedio. Estos grupos se conocen como *manadas* y se rigen por medio de una estricta pirámide jerárquica, como se muestra en la Figura 2.15.

En la punta de la pirámide se encuentra el lobo *alfa*, quien es el lobo líder de la manada y el principal responsable en la toma de decisiones. Sus órdenes deben de ser seguidas por los otros miembros del grupo y este puesto suele ser ocupado por el mejor en términos de gestión de la manada. En la segunda posición, se encuentran los lobos *beta*, los cuales son los subordinados y consejeros del alfa, ya que ayudan en la toma de decisiones y es muy probable que uno de estos lobos sea el más calificado para reemplazar al *alfa* en caso de ser necesario. En el tercer lugar jerárquico se encuentran los lobos *delta* encargados de los trabajos básicos como exploración, caza y vigilancia, así como encargarse de la tarea de proteger a la manada. Finalmente, en la base de la pirámide se encuentran los lobos *omega* quienes son los renegados del grupo. En el algoritmo se trata de emular este sistema jerárquico, de modo que la posición del *alfa*, *beta* y *delta* se le asigna a la primera, segunda y tercera solución más apta respectivamente.

Para la búsqueda de soluciones, este algoritmo se basa en el comportamiento de los lobos al momento de conseguir comida. Los lobos grises emplean un comportamiento de caza en grupo para la cual se dedican al rastreo, persecución y acecho constante a la presa para desgastarla físicamente. Rodear a la presa evita que esta pueda escapar y atacar a una presa cansada aumenta las probabilidades de éxito.

El GWO aborda un problema de optimización por búsqueda de forma similar a la que los lobos buscan a su presa. En este aspecto, como ya se explicó anteriormente, las tres soluciones con el valor de aptitud más alto obtiene las jerarquías mayores, mientras que los demás individuos de la población se usan únicamente como agentes de búsqueda. Para emular el comportamiento envolvente de la caza de los lobos se utiliza la Ecuación 2.10 y 2.11.

$$\vec{D} = |\vec{C} * \vec{X}_p(t) - \vec{X}(t)| \tag{2.10}$$

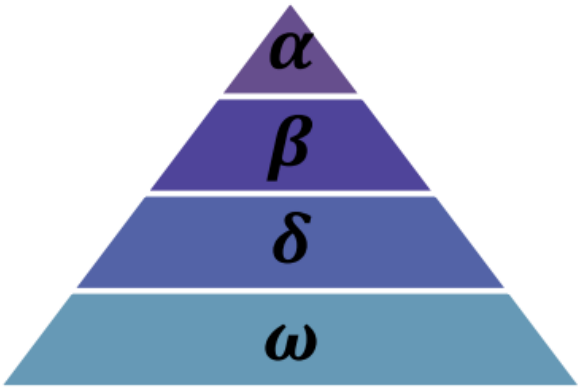


Figura 2.15: Jerarquía de lobo gris. Fuente: (Mirjalili et al., 2014)

$$\vec{X}(t+1) = \vec{X}_p(t) - \vec{A} * \vec{D} \tag{2.11}$$

donde D es una distancia, t representa la iteración actual, X_p es el vector de posición de la presa, X indica la posición del lobo gris.

Por último A y C son coeficientes que se pueden calcular con las Ecuaciones 2.12 y 2.13.

$$\vec{A} = 2\vec{a} * \vec{r}_1 - \vec{a} \tag{2.12}$$

$$\vec{C} = 2 * \vec{r}_2 \tag{2.13}$$

donde $\vec{r}_1$ y $\vec{r}_2$ son vectores aleatorios en el rango de $[0, 1]$, mientras que el valor de a se reduce de manera lineal de 2 a 0 según el pasar de las iteraciones.

El coeficiente A determina el comportamiento de los agentes de búsqueda, y su rango de valores va de $[-2a, 2a]$, con valores mayores a 1 o menores a -1 para ayudar al agente a converger sobre la presa. Por su parte, C ayuda a mostrar un comportamiento más aleatorio durante la optimización y evitar el estancamiento en óptimos locales. Con estas ecuaciones se simula el comportamiento envolvente de los lobos por lo que, la posición del lobo se encuentra en una ubicación aleatoria alrededor de la presa. Esto se puede observar de forma gráfica en la Figura 2.16.

En este modelo, la búsqueda se ve guiada principalmente por las tres mejores soluciones, las cuales se utilizan para estimar la posición de la presa y obtener un nuevo vector de posición según las ecuaciones siguientes:

$$\vec{D}_\alpha = |\vec{C}_1 * \vec{X}_\alpha - \vec{X}|, \;\; \vec{D}_\beta = |\vec{C}_2 * \vec{X}_\beta - \vec{X}|, \;\; \vec{D}_\delta = |\vec{C}_3 * \vec{X}_\delta - \vec{X}| \tag{2.14}$$

$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}_1 * \vec{D}_\alpha, \;\; \vec{X}_2 = \vec{X}_\beta - \vec{A}_2 * \vec{D}_\beta, \;\; \vec{X}_3 = \vec{X}_\delta - \vec{A}_3 * \vec{D}_\delta \tag{2.15}$$

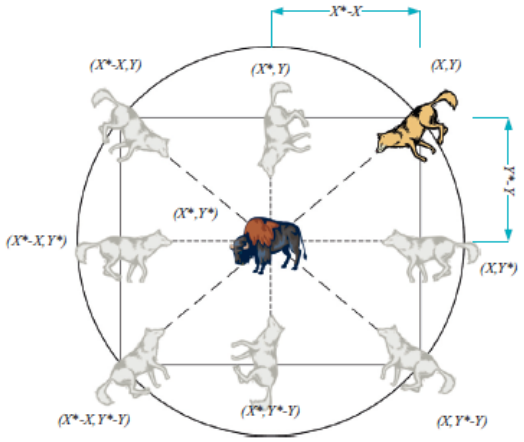


Figura 2.16: Comportamiento envolvente de los lobos. Fuente: (Mirjalili et al., 2014)

$$\vec{X}(t+1) = \frac{\vec{X}_1 + \vec{X}_2 + \vec{X}_3}{3} \tag{2.16}$$

En la Ecuación 2.14 se calculan las direcciones $\vec{D}$ de las tres mejores soluciones utilizando el coeficiente C, la posición actual de cada una de éstas y su respectiva posición general $\vec{X}$. En la Ecuación 2.15 se calcula la nueva posición de cada una de las tres soluciones utilizando la Ecuación 2.12 para cada una. Finalmente, en la Ecuación 2.16 se calcula la nueva posición general. En cada una de las iteraciones, se calcula la aptitud de cada agente de búsqueda y nuevamente se seleccionarán a las tres mejores soluciones, es decir a los nuevos lobos alfa, beta y delta, para que guíen la nueva búsqueda.

Capítulo 3

Desarrollo del proyecto

En este capítulo se describe el funcionamiento así como la metodología utilizada para la construcción de cada uno de los módulos del proyecto de clasificación de COVID-19 y la optimización de hiperparámetros mediante metaheurísticas. En la primera sección, se describen las características del hardware y software utilizados. En la segunda sección, se presenta la propuesta de optimización de hiperparámetros, así como una descripción de cada uno de los módulos que componen este trabajo de tesis.

3.1. Especificaciones de hardware y software

Los experimentos reportados en este trabajo de tesis se realizaron en el lenguaje de programación `Python Versión 3.9`. Además, se utilizaron únicamente bibliotecas de acceso abierto como herramientas para el desarrollo de los módulos. Las diferentes arquitecturas de red CNN utilizadas así como el entrenamiento y evaluación de los mismos se implementaron usando las bibliotecas de `Keras` y `Tensorflow` así como sus respectivas dependencias. Por otra parte, `OpenCV` proporciona herramientas que facilitan el preprocesamiento de las imágenes. Finalmente, las bibliotecas `Imblearn` y `Sklearn` proporcionan una mejor visualización de los resultados por medio de matrices de confusión y diversas medidas de rendimiento.

En términos del hardware, todos los experimentos fueron ejecutados en un servidor, el cual cuenta con las siguientes características: CPU Intel(R) Xeon(R) E5-2630 de 2.40GHz de 32 núcleos, sistema operativo Ubuntu 18.04 LTS, un disco de estado sólido de 120 GB, un disco duro de 1TB, memoria RAM DDR4 de 16 GB y una GPU NVIDIA Tesla K40c con capacidad de cómputo 3.5.

3.2. Módulos del proyecto

El presente trabajo de tesis consta de dos módulos principales, cada uno con una tarea específica. En primer lugar, se encuentra el *módulo de preprocesamiento*, en el cual se realiza

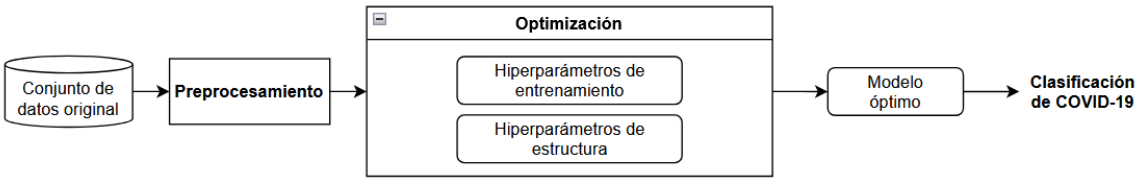


Figura 3.1: Esquema general del proyecto

el tratamiento de los conjuntos de datos utilizados. El segundo es el *módulo de optimización*, el cual se presenta como una propuesta de estrategia de optimización de hiperparámetros para CNNs, con la cual se realiza la búsqueda de los valores óptimos por medio de la experimentación con las tres metaheurísticas planteadas en esta tesis: *algoritmos genéticos*, *evolución diferencial* y *optimización de lobos grises*. Finalmente, se encuentra la clasificación de COVID-19, es decir, se utilizan los datos preprocesados en el módulo uno y los hiperparámetros obtenidos en el módulo dos para realizar la clasificación de COVID-19. El esquema del proyecto se muestra en la Figura 3.1.

3.2.1. Módulo de preprocesamiento

Antes de comenzar con la experimentación, es necesario realizar un tratamiento sobre las imágenes de los diferentes conjuntos de datos. El preprocesamiento de los datos es necesario, ya que algunos de los conjuntos están conformados por imágenes de diversas fuentes, lo que se puede traducir en datos no homogéneos aun para un mismo conjunto de datos. De igual manera, es necesario redimensionar los datos de los diferentes conjuntos para que correspondan con la entrada de los modelos CNN durante la experimentación.

Para lograrlo, se aplica a cada uno de los conjuntos de datos utilizados el preprocesamiento mostrado en la Figura 3.2. Primero, el conjunto de datos es dividido en tres subconjuntos: entrenamiento, validación y prueba, siguiendo la distribución mostrada en la Figura 3.3. Después, dependiendo de las características del conjunto utilizado se decide si realizar o no aumento de datos sobre los conjuntos de entrenamiento y validación. Finalmente, a todas las imágenes resultantes se le aplica un redimensionamiento y normalización. A continuación, se describen las técnicas de preprocesamiento utilizadas.

Aumento de datos

Se decidió aplicar *aumento de datos* debido a que se trabaja con conjuntos de datos que no cuentan con una gran cantidad de imágenes por clase o que están conformados por clases desbalanceadas. El aumento de datos se considera una tarea opcional según el conjunto en cuestión, ya que no todos los conjuntos de datos utilizados presentan estos problemas. Además, como se observa en el esquema de la Figura 3.2, el aumento de datos es

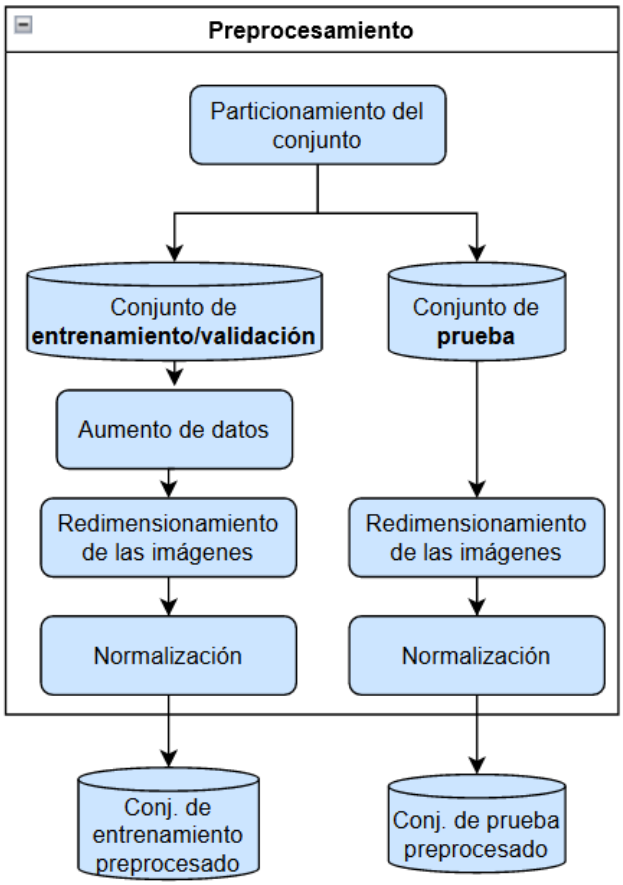


Figura 3.2: Esquema del módulo de preprocesamiento

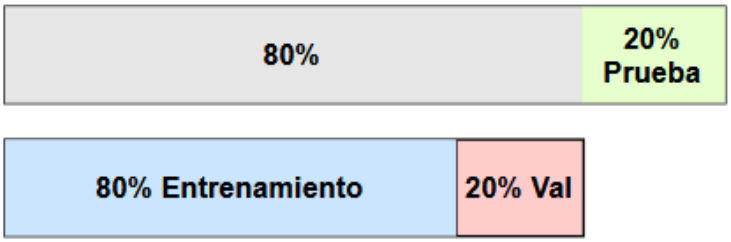


Figura 3.3: Distribución de los conjuntos de entrenamiento, validación y prueba

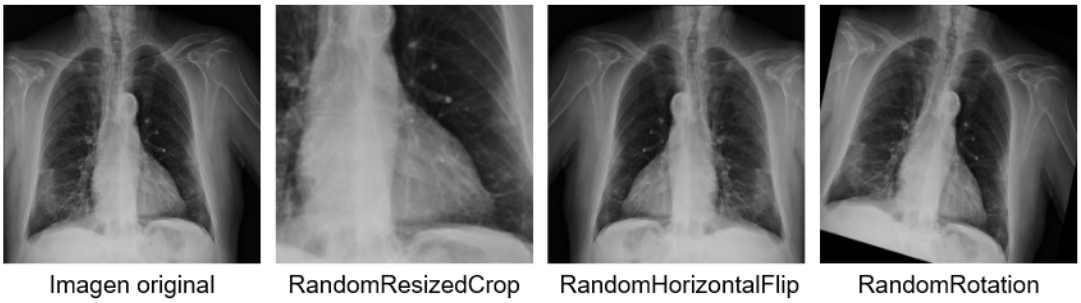


Figura 3.4: Transformaciones utilizadas para el aumento de datos

aplicado únicamente sobre las particiones de entrenamiento y validación del conjunto de datos original. Realizar el aumento de datos a un conjunto de imágenes permite generar nuevas imágenes a partir de las originales aplicando a cada elemento una o varias transformaciones. De esta manera, se puede incrementar el número de elementos de una clase en específico para balancear el conjunto y/o incrementar el número de imágenes de todas las clases cuando el conjunto de datos es muy pequeño (Shorten and Khoshgoftaar, 2019).

Este método se implementó siguiendo el esquema presentado en el trabajo de Lopez-Betancur et al. (2021), donde se utiliza una secuencia de tres transformaciones para generar las nuevas imágenes. Estas transformaciones son:

- *RandomResizedCrop*. Consiste en realizar un recorte en la imagen con un área y una relación de aspecto aleatorios para posteriormente redimensionar el recorte a un tamaño determinado.

- *RandomHorizontalFlip*. Consiste en generar, con una probabilidad de 50 %, una reflexión horizontal de la imagen de entrada.

- *RandomRotation*. Consiste en rotar la imagen de entrada con un ángulo aleatorio comprendido en un intervalo. El intervalo para este trabajo fue definido entre 0 y 20 grados.

Al tener una imagen como entrada, se le aplica esta secuencia de transformaciones obteniendo una única imagen de salida, la cual estará transformada con cualquier combinación posible de los parámetros antes mencionados. En la Figura 3.4 se muestra el efecto de las transformaciones utilizadas en el aumento de datos.

Redimensionamiento de las imágenes

Los conjuntos de datos utilizados están conformados por imágenes de dimensiones variables, especialmente si los conjuntos fueron construidos con datos de diversas fuentes. En general, un modelo de CNN asume una imagen de entrada con dimensiones predefinidas,

lo que significa que el alto y ancho de las imágenes que se utilizan así como el número de canales están definidos por el modelo en cuestión. Es esta la principal razón por la que el redimensionamiento de las imágenes es una tarea esencial.

El redimensionamiento se aplica a todas las imágenes de los conjuntos utilizados, reduciendo el ancho y alto de las mismas a un tamaño de 225×225. Estas dimensiones son generales en todos los experimentos debido a que suele ser un tamaño de imagen que ofrece buenos resultados en las operaciones de convolución, siendo utilizada en diferentes arquitecturas de red predefinidas como las redes residuales.

Normalización

Las imágenes que conforman los conjuntos de datos utilizados corresponden a imágenes de 8-bits, para aquellas que se encuentran en escala de grises, y de 24-bits para las que tienen formato RGB. En cualquier caso, los niveles de intensidad están dados por valores enteros entre 0 y 255, los cuales se encuentran en una escala muy diferente a la escala con la que normalmente trabajan la mayoría de modelos de DNN. Por esta razón, se decidió normalizar el rango de intensidad de las imágenes en una escala entre 0 y 1 multiplicando los datos por un factor de 1/255.

3.2.2. Módulo de optimización

El siguiente paso en la implementación de este proyecto es la optimización de los hiperparámetros de una DNN por medio de metaheurísticas. Esta propuesta consiste en realizar el ajuste de los valores de los hiperparámetros de una CNN y encontrar una combinación de los mismos que proporcione una mejora en el rendimiento de la red en cuanto a clasificación de COVID-19 se refiere.

En este módulo se definen los hiperparámetros a optimizar, los cuales componen a los individuos o agentes de búsqueda en los algoritmos de optimización presentados en esta tesis en la Sección 2.4.1. Además, se utilizan las metaheurísticas para generar combinaciones de estos hiperparámetros, las cuales se consideran como candidatos para generar un modelo DNN óptimo para la clasificación de COVID-19 usando imágenes de XR. El esquema particular para el módulo de optimización se presenta en la Figura 3.5, en donde se muestran las tres metaheurísticas utilizadas para la optimización, cada una de ellas se ejecuta de manera independiente pero siguen el mismo proceso para la evaluación de los individuos. Cada uno de los modelos generados es entrenado con el conjunto de entrenamiento y la evaluación de la función objetivo está dada en función de su rendimiento de clasificación, específicamente, según la exactitud obtenida con el conjunto de validación. Al finalizar la optimización, se obtienen los hiperparámetros que se utilizan para realizar la clasificación de COVID-19.

El comportamiento general del proceso de optimización de los hiperparámetros se presenta en el Algoritmo 1. La optimización de los hiperparámetros se realiza mediante una búsqueda metaheurística declarada como la función $Meta$, la cual representa a cualquiera de

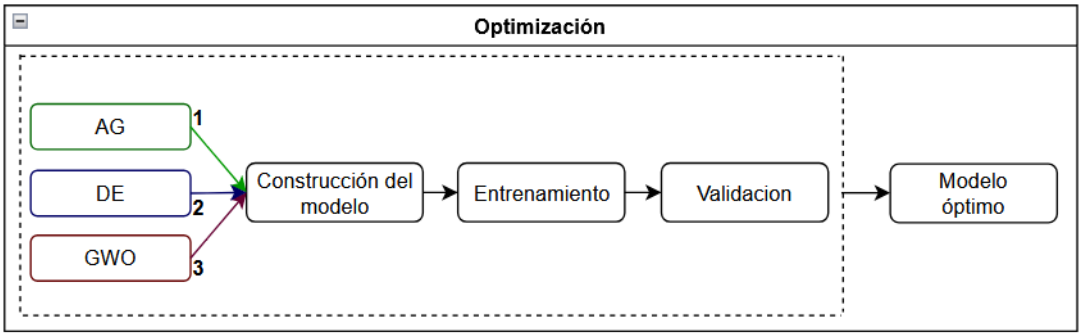


Figura 3.5: Esquema del módulo de optimización

las tres metaheurísticas seleccionadas para este proyecto. Los conjuntos de entrenamiento y validación son utilizados en este proceso para obtener la aptitud de cada individuo generado durante la optimización. Ésto se hace mediante la función objetivo $FuncObj$, donde cada individuo i de la población P es entrenado con el conjunto X_T y su aptitud está dada por la exactitud de clasificación A_i obtenida respecto al conjunto X_V.

Algoritmo 1: Algoritmo de optimización con metaheurísticas

$X_T \leftarrow$ Conjunto de muestras de entrenamiento
$X_V \leftarrow$ Conjunto de muestras de validación

Obtener el mejor modelo $M_{Max} \leftarrow Meta(X_T, X_V, FuncObj)$

Función *FuncObj(P)* **:**
  para $i \in P$ **hacer**
    Construcción/inicialización del modelo M_i
    Entrenar modelo M_i usando el conjunto X_T
    Obtener la aptitud A_i del modelo M_i utilizando el conjunto X_V
  fin
  return A
fin

Optimización de hiperparámetros

Antes de iniciar con la optimización es necesario definir cuáles son los hiperparámetros de la red sobre los cuales se realizará el ajuste y los límites de búsqueda definidos para cada uno. Para ésto, se realizan dos implementaciones: optimización de hiperparámetros de entrenamiento y optimización de hiperparámetros de estructura.

Hiperparámetro	Frecuencia/total
LR	5/7
BS	5/7
Épocas	3/7

Cuadro 3.1: Hiperparámetros de entrenamiento seleccionados para la optimización

Hiperparámetros de entrenamiento y codificación del individuo

La primera de estas implementaciones consiste en realizar únicamente la optimización de hiperparámetros de entrenamiento, utilizando una estructura CNN predefinida como base para realizar la clasificación de COVID-19. La elección de los hiperparámetros de entrenamiento a utilizar se llevó a cabo mediante una investigación de la literatura, tomando en consideración siete trabajos en los que se realiza ajuste y optimización de hiperparámetros de entrenamiento de una CNN (Ahmad et al., 2021; Irmak, 2021; Khan et al., 2023; Kiziloluk and Sert, 2022; Goel et al., 2021; Serizawa and Fujita, 2020; Nagib et al., 2022). En dicha revisión, se encontró el uso de 10 diferentes hiperparámetros de entrenamiento, de los cuales sólo se seleccionaron los tres más utilizados frecuentemente. Dicha revisión se encuentra en el Cuadro 3.1, en el que se muestran los hiperparámetros de entrenamiento seleccionados y su frecuencia de aparición.

Tomando en consideración la cantidad y los hiperparámetros más empleados en la literatura, se seleccionan los hiperparámetros de entrenamiento. De esta manera, los individuos o agentes de búsqueda para cada una de las metaheurísticas empleadas en este trabajo de tesis quedan representados por un vector de tres elementos o genes, donde cada elemento representa un hiperparámetro de entrenamiento (LR, BS y número de épocas). Además, se utiliza un espacio de búsqueda amplio para cada hiperparámetro, ésto con la intención de obtener un mejor rendimiento durante la optimización. En la Figura 3.6, se muestra la representación del individuo para los hiperparámetros de entrenamiento, así como su respectivo espacio de búsqueda.

Hiperparámetros de estructura y codificación del individuo

La segunda implementación de la optimización toma en consideración tanto hiperparámetros de entrenamiento como de estructura, dando principal importancia a estos últimos. Para ésto, se toma como referencia el esquema de optimización presentado en el trabajo de Nagib et al. (2022), del cual se seleccionaron los siguientes hiperparámetros para optimizar.

- Número de capas convolucionales. No se utiliza una estructura o arquitectura previamente definida de red para realizar la clasificación; en su lugar, se prueban estructuras

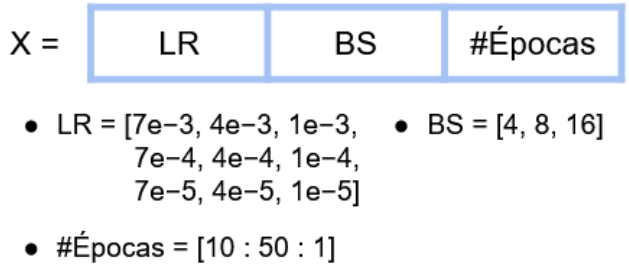


Figura 3.6: Codificación del individuo para los hiperparámetros de entrenamiento

generadas durante la optimización con diferente cantidad de capas convolucionales. Uno de los objetivos de la optimización de la estructura es obtener una red ligera, por lo que se establece un limite de 7 capas como máximo y un mínimo de 1 capa convolucional.

- Número de filtros. Se pretende encontrar un número de filtros adecuado para cada capa convolucional, según la estructura generada. En el caso de las capas densas, este número hace referencia a la cantidad de neuronas de cada capa y está dado por un valor entero en un rango de 1 a 512.

- Tamaño de lote. Un hiperparámetro de entrenamiento que determina el número de muestras que se propagan en el modelo de red antes de realizar la retropropagacion del error y actualizar pesos. Definido como un número entero entre 4 y 32.

- Número de épocas. Una época indica que todos los elementos del conjunto de entrenamiento han sido vistos o propagados en la red, así como también se ha retropropagado su error correspondiente. Entonces, el número de épocas es una cantidad que determina el número de iteraciones o veces que el conjunto de entrenamiento completo ha sido visto por la red. En nuestra experimentación se establece un mínimo de 25 y un máximo de 150 épocas.

Adicional a esto, en lugar de utilizar únicamente capas convolucionales para generar la estructura de la red, se propone experimentar con los tres siguientes enfoques para la construcción de la red.

- Modelo convolucional básico. Corresponde a una estructura CNN dividida en dos partes. El cuerpo de la red, conformado por bloques convolucionales básicos, que a su vez se componen de capas *Conv2D* con filtros de tamaño 3×3, una capa de activación *ReLU* y una capa *MaxPool2D* con kernel de tamaño 2×2. La segunda parte del modelo corresponde a la parte de clasificación y consta de las capas completamente

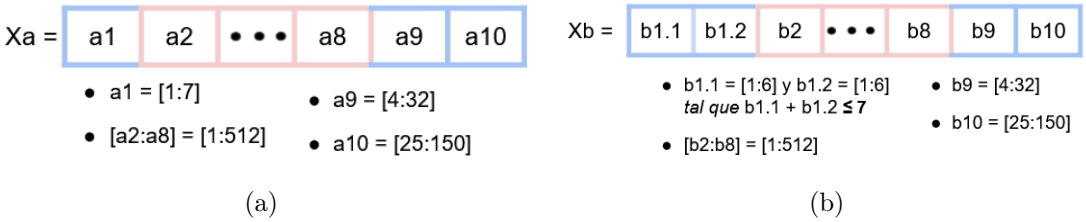
conectadas finalizando con la capa de salida. Para este modelo, tanto el número de bloques convolucionales básicos como el total de capas densas, se consideran parámetros a optimizar.

- Modelo multiescala. Inspirado en la red MFL propuesta en el trabajo de Joshi and Nayak (2022) (ver Sección 2.3.1), éste modelo consta de tres partes. La primera es la cabecera y contiene a la capa de entrada, un *mini_ block MFL* y una capa *MaxPool2D* con kernel de tamaño 2×2. En seguida, se encuentra el cuerpo de la red conformado por los bloques MFL, cada uno seguido de una capa *MaxPool2D* de 2×2 a excepción del último bloque. Finalmente, se tiene la parte de clasificación donde, a diferencia del modelo anterior, no se cuenta con capas densas sino que en su lugar se utiliza una capa *GlobalAveragePooling2D* seguida de una capa de abandono con probabilidad del 50 % y finalmente la capa de salida. Dado que no se utilizan capas densas, únicamente la cantidad de bloques MFL del cuerpo del modelo es considerada como parámetro a optimizar.

- Modelo residual. Utiliza los bloques residuales de cuello de botella presentados en el trabajo de He et al. (2016) (ver Sección 2.3.1). Al igual que el modelo anterior, se compone de tres partes. La cabecera mantiene la misma estructura que en las arquitecturas *ResNet*, es decir, la capa de entrada seguida de una capa *Conv2D* con 64 filtros de tamaño 7×7 y una capa *MaxPool2D* con kernel de tamaño 3×3. El cuerpo está conformado únicamente por bloques residuales de cuello de botella. Por último, el apartado de clasificación está dado por una capa *AdaptiveAveragePooling2D* y la capa de salida. En este modelo únicamente el número de bloques de cuello de botella es considerado como un parámetro a optimizar generando estructuras que contienen entre 1 y 7 bloques residuales.

Los hiperparámetros antes descritos conforman la estructura de los diferentes individuos o agentes de búsqueda utilizados para cada uno de los algoritmos de optimización. Además, se aclara que los valores mínimos y máximos de los hiperparámetros están relacionados directamente con el hardware descrito en la Sección 3.1. Dado que el modelo convolucional básico propuesto contempla la optimización de capas densas, además de los bloques convolucionales, es necesario definir la construcción de un individuo específico para dicho modelo. En la Figura 3.7 se presentan los dos tipos de individuos propuestos para los hiperparámetros de estructura.

Observe en la Figura 3.7a que $a1$ representa el número de bloques convolucionales; los genes desde $a2$ hasta $a8$ representan la cantidad de filtros o neuronas que tendrá cada una de los potenciales bloques convolucionales de la red dados por $a1$; $a9$ corresponde al tamaño de lote y $a10$ es la cantidad de épocas de entrenamiento. De la misma manera, en el caso del modelo convolucional básico, representado por la Figura 3.7b, $a1$ se subdivide en $b1{,}1$ y $b1{,}2$ representando el número de bloques convolucionales y capas densas respectivamente. En



(a) (b)

Figura 3.7: Codificación de los 2 individuos para los hiperparámetros de estructura

este caso, es necesario agregar una restricción, de modo que aquellos individuos cuya suma de $b1{,}1$ y $b1{,}2$ sea mayor a 7 no serán considerados como individuos válidos.

Experimentos

Para obtener un rendimiento satisfactorio en el proceso de optimización se decidió realizar dicho proceso por medio de tres experimentos diferentes. Tales experimentos se detallan a continuación:

- Búsqueda gruesa (BG). Consiste en utilizar un espacio de búsqueda amplio para los valores de todos los hiperparámetros a optimizar y realizar la optimización por un total de 6 iteraciones con una población de 15 individuos.

- Búsqueda Fina (BF). Funciona como un refinamiento de la BG al realizar la optimización de los hiperparámetros en un espacio de búsqueda más reducido. Esta búsqueda se realiza alrededor de los valores obtenidos durante la BG, por lo que el espacio de búsqueda depende en su totalidad de los resultados obtenidos en el primer experimento. Además, para este experimento se utiliza el mismo tamaño de población y la misma cantidad de iteraciones que en la BG.

- Búsqueda extensa (BE). Es independiente de los anteriores y consiste en realizar la optimización en el mismo espacio de búsqueda que el utilizado en la BG pero con mayor diversidad de información, específicamente utilizando 10 iteraciones y una población de 30 individuos. Cabe señalar que, debido al alto costo computacional que implica evaluar una DNN, no se seleccionó una cantidad de individuos más amplia.

Cabe aclarar que los experimentos presentados anteriormente se toman en consideración tanto para la optimización de hiperparámetros de entrenamiento como para los de estructura. Asimismo, el tamaño de la población y número de iteraciones especificado en cada uno es general. Es decir, se utiliza para cada una de las metaheurísticas a utilizar.

Configuración de las metaheurísticas

La codificación de los individuos respectiva de cada implementación de la optimización es compatible con las tres metaheurísticas a utilizar (GA, DE, GWO), por lo que los individuos generados que conforman a la población de cada una de éstas sigue la misma estructura presentada en las Figuras 3.6 y 3.7. Asimismo, el tipo de experimento a realizar especifica el tamaño de la población y el número de iteraciones de la optimización. Otros parámetros fijos independientes del experimento, el enfoque y la metaheurística utilizada son: activación *softmax* para las capas de salida, optimizador de Descenso de Gradiente Estocástico (SGD, por sus siglas en inglés) y función de pérdida de entropía de cruce categórica (*categorical_ crossentropy*).

Las tres metaheurísticas anteriormente mencionadas hacen uso de una función objetivo, la cual se encarga de evaluar a cada uno de los individuos generados durante la optimización y proporcionar una calificación a éstos en forma de su valor de aptitud. El valor de aptitud de cada individuo es importante para identificar a aquellos que pueden considerarse como candidatos a solución o, en este caso, determinar la mejor combinación de hiperparámetros. Para este trabajo, las tres metaheurísticas mencionadas utilizan la misma función objetivo. La función objetivo se compone de tres partes principales que son, la construcción del modelo, el entrenamiento y la validación de la red.

- Construcción del modelo. Consiste en realizar el tratamiento necesario a los individuos para poder utilizar sus respectivos hiperparámetros de manera correcta. En el caso de los hiperparámetros de estructura, es en esta fase donde se genera la estructura de red especificada por el individuo correspondiente. Para los hiperparámetros de entrenamiento, es aquí donde se invoca el modelo de la red predefinida a utilizar.

- Entrenamiento. Recibe como parámetros el modelo definido en la fase anterior, la función de pérdida, el optimizador seleccionado, el número de épocas y el subconjunto de entrenamiento. El entrenamiento se realiza en un ciclo iterativo determinado por el número de épocas, en el cual los datos de entrada son pasados al modelo por lotes. Esto genera una salida, la cual es comparada con la salida real, dada por las etiquetas de clase del conjunto; de esta manera el error es calculado por la función de pérdida y se procede con el ajuste de los pesos. Para cada época de entrenamiento, se calcula la pérdida y exactitud del modelo y se muestran en consola para proporcionar un seguimiento visual del comportamiento del entrenamiento del modelo. Una vez finalizado el entrenamiento, se obtiene el nuevo modelo con los pesos ajustados.

- Validación. Recibe como parámetros el modelo con los pesos ajustados obtenido en la fase de entrenamiento, la función de pérdida y el conjunto de validación. La validación se realiza en una única iteración en la que todos los datos del conjunto de validación son enviados como entrada al modelo y se obtiene su respectiva predicción. Finalmente, la validación regresa la exactitud obtenida por el modelo respecto al conjunto de

validación. Esta misma medida de rendimiento es considerada como el resultado de la función objetivo y la aptitud del individuo correspondiente.

Lo anteriormente mencionado corresponde a configuraciones generales, utilizadas por las tres metaheurísticas. El flujo de trabajo de cada metaheurística y sus parámetros específicos se explican a continuación.

Algoritmo genético simple con elitismo

La primera implementación de la optimización de los hiperparámetros es mediante el uso de GA. El funcionamiento general y la teoría tras esta metaheurística se explica más a detalle en la sección 2.4.1. Para adaptar este algoritmo a nuestro problema de optimización se siguió el esquema presentado en la Figura 3.8.

La creación de la población inicial se realiza siguiendo un enfoque aleatorio, es decir, cada uno de los individuos que conforman la población inicial es generado al asignar, a cada uno de los genes que lo conforma, un número aleatorio dentro de los limites respectivos. Las generaciones posteriores son generadas por medio de la recombinación genética siguiendo el esquema del algoritmo. La aptitud de cada individuo de la población es calculada mediante la función objetivo y está dada en términos de la exactitud obtenida al evaluar el modelo utilizando el conjunto de validación.

La nueva población es generada en un proceso de cuatro pasos. Primero, se realiza la selección de cada par de padres mediante el operador de *selección por torneo* con cuatro competidores, el cual consiste en seleccionar de manera aleatoria dos parejas de individuos de la población actual. Los individuos más aptos de cada pareja serán elegidos como los padres. El segundo paso consiste en realizar la cruza de los padres seleccionados mediante el operador de *cruza uniforme* con probabilidad de 50 %. Para evitar los mínimos locales se implementa un operador de *mutación aleatoria mediante remplazo* con una probabilidad de 10 %. Finalmente, se utiliza una estrategia de *elitismo*, con la que se elige conservar al mejor individuo de la población actual que, sumados a los hijos generados, conforman la nueva población a evaluar.

Evolución diferencial

El algoritmo DE se implementó utilizando la biblioteca `scipy`. Esta biblioteca contiene un módulo de optimización que brinda acceso a una variedad de algoritmos de minimización de funciones objetivo entre las que se encuentra el algoritmo de evolución diferencial. La implementación de evolución diferencial que se encuentra en la biblioteca es exactamente la misma que la de la literatura presentada en la Sección 2.4.1 y sigue un esquema similar al de la Figura 3.9. La ventaja de la biblioteca es que permite elegir entre diferentes estrategias para la selección de candidatos para realizar la mutación, además de que optimiza la ejecución de los cálculos y permite utilizar varios subprocesos para reducir el tiempo de ejecución.

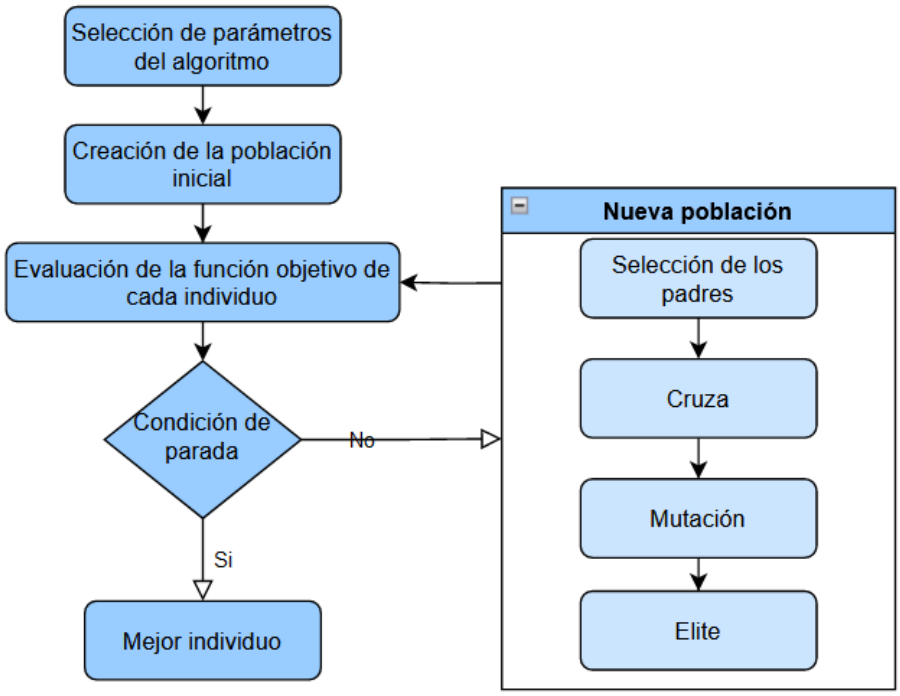


Figura 3.8: Esquema de la optimización con algoritmos evolutivos

Un punto importante de la biblioteca `scipy` es que considera al algoritmo DE como un problema de minimización y ya que el objetivo en este experimento es maximizar el rendimiento de clasificación, es necesario representar el resultado de la función objetivo como un número negativo. Otro parámetro a modificar es la tolerancia, la cual representa la condición de parada. Sin embargo, no se toma en consideración ya que se desea que la ejecución del algoritmo finalice al completar todas las iteraciones.

Para el caso de los parámetros específicos del algoritmo DE se utilizaron los valores por defectos declarados para la función *differential_evolution*. Esto es, un factor de mutación F con un valor aleatorio entre 0,5 y 1, una constante de recombinación C_r de 0,7 y una inicialización *latinhypercube*. Por último, se utiliza una estrategia de recombinación *best1bin* para la creación de los nuevos candidatos de prueba.

Optimización con lobos grises

El algoritmo GWO utilizado para la optimización se obtuvo de la biblioteca `mealpy` y corresponde al presentado en la Sección 2.4.1. Para efectos del problema abordado en este trabajo de tesis, se sigue el esquema mostrado en la la Figura 3.10. Los agentes de búsqueda están representados por un vector de valores que corresponden a los hiperparámetros de red a optimizar. La posición inicial de cada uno de estos agentes se obtiene asignando un valor

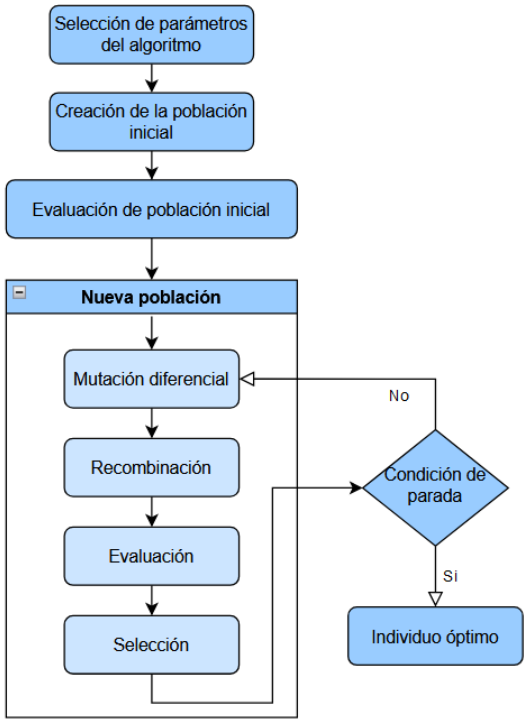


Figura 3.9: Esquema de la optimización con evolución diferencial

aleatorio a cada elemento del vector según los límites correspondientes.

En cada iteración, se establece una nueva jerarquía de los agentes de búsqueda. Para hacer esto de manera sencilla, simplemente se ordenan todos los agentes de acuerdo a su exactitud obtenida en la evaluación. Los tres agentes de búsqueda que obtienen la puntuación más alta son designados como los lobos alpha, beta y delta respectivamente.

Como se explica en la Sección 2.4.1, la posición $\vec{X}_\alpha$, $\vec{X}_\beta$ y $\vec{X}_\delta$ de las tres mejores soluciones está representada por el vector de hiperparámetros. Estos vectores son utilizados para obtener las direcciones $\vec{D}_\alpha$, $\vec{D}_\beta$ y $\vec{D}_\delta$, cada una de las cuales determinan la nueva posición de las tres mejores soluciones tomando como partida la posición actual de cada lobo. El algoritmo GWO considera que, los agentes de búsqueda siempre se mueven en dirección de la presa tratando de rodearla; por lo tanto, la posición de la presa en cada iteración está dada por el promedio de los vectores de hiperparámetros de las tres mejores soluciones en la iteración actual.

Al igual que en las otras metaheurísticas, la condición de parada está dada por el total de iteraciones definidas para la ejecución del algoritmo, ignorando la implementación de algún valor de tolerancia para detener la ejecución antes de tiempo.

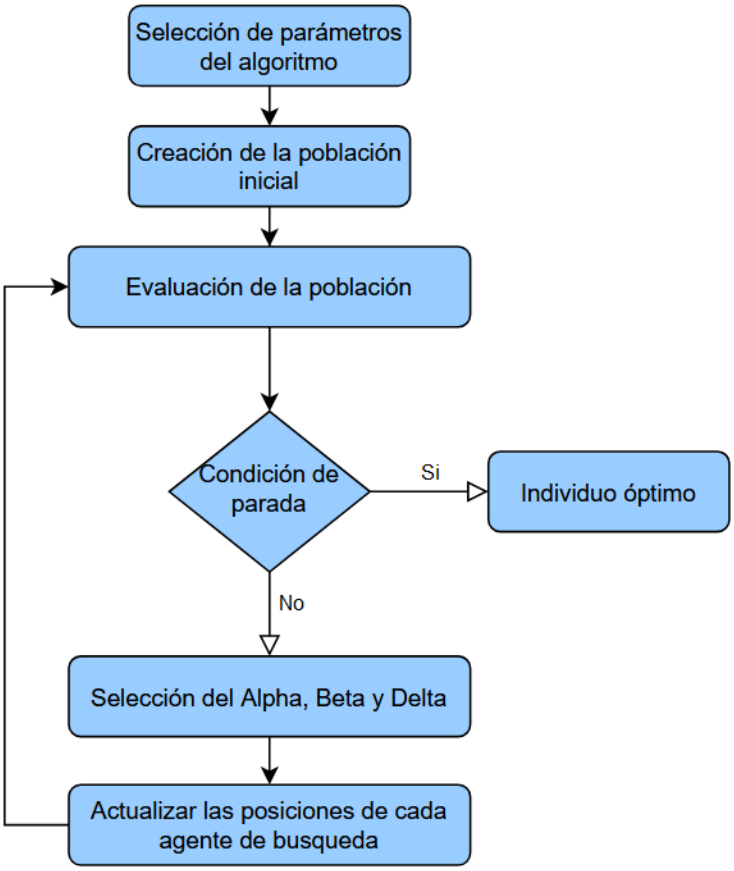


Figura 3.10: Esquema de optimización con lobos grises

Capítulo 4

Resultados

En este capítulo se presentan los resultados obtenidos como producto de la investigación y experimentación del presente trabajo de tesis. En la primera sección, se reportan las bases de datos utilizadas para realizar la clasificación de COVID-19 en los diferentes experimentos realizados. En la segunda sección, se reportan las medidas de rendimiento utilizadas para evaluar el rendimiento de clasificación. En la tercera sección, se muestran los resultados del rendimiento de clasificación obtenidos con diferentes arquitecturas CNNs predefinidas, para cada uno de los conjuntos de datos. En la cuarta sección, se muestran los resultados correspondientes a la optimización de hiperparámetros de entrenamiento sobre una estructura CNN predefinida. En la última sección, se presentan los resultados obtenidos con la optimización de hiperparámetros de estructura.

4.1. Conjuntos de datos

Para llevar a cabo la fase experimental de este trabajo de tesis, se utilizan dos conjuntos de datos. El primero, al cual llamaremos *Chest+COVID-19*, consta de imágenes pertenecientes a 3 clases: COVID-19, Neumonía y Normal, las cuales fueron obtenidas de dos conjuntos de datos encontrados en la literatura. Las imágenes de la clase COVID-19 fueron obtenidas del conjunto *Covid-chestxray* (Cohen et al., 2020), filtrando únicamente las imágenes de rayos X de tórax de vista frontal de pacientes con un diagnóstico positivo a la prueba RT-PCR, obteniendo un total de 117 datos. Para el caso de la clase Neumonía y Normal, las imágenes se obtuvieron del conjunto de datos *Chest X-Ray Images (Pneumonia)* (Kaggle, 2018), el cual contiene un total de 5856 imágenes, de las cuales 4273 pertenecen a la clase Neumonía y las 1583 restantes corresponden a pacientes sanos. Dado que se trata de un conjunto de datos desbalanceado, se hace uso del aumento de datos presentado en la Sección 3.2.1, obteniendo la distribución del conjunto aumentado presentado en el Cuadro 4.1.

El conjunto *Chest+COVID-19* presenta algunas falencias típicas de los conjuntos de datos tradicionales de COVID-19, tales como: datos sesgados a casos de gravedad, etiquetas

	Entrenamiento			Prueba			
Conjunto	**COVID-19**	**Normal**	**Neumonía**	**COVID-19**	**Normal**	**Neumonía**	**Total**
Original	94	1273	3418	23	310	855	5973
Aumentado	3478	3819	3418	23	310	855	11903

Cuadro 4.1: Distribución del conjunto de datos *Chest+COVID-19*

y señales de imagen tanto fuera como dentro de la zona pulmonar y datos no homogéneos, una consecuencia directa de construir dichos conjuntos mediante datos obtenidos de diversas fuentes. Para evitar estos problemas e incrementar la aceptabilidad de los modelos generados, se decidió utilizar también el conjunto de datos COVIDGR (Tabik et al., 2020). Este es definido como un conjunto de alta calidad con datos recolectados en el Hospital Universitario Clínico San Cecilio de Granada, España. Contiene únicamente radiografías de tórax de vista postero-anterior y las imágenes no cuentan con marcas ni etiquetas cerca de la región pulmonar. La evaluación de las imágenes fue realizada por cuatro expertos y el etiquetado de la clase COVID-19 se establece como positivo sólo si la prueba RT-PCR y el radiólogo lo confirman. En cuanto a su distribución, cuenta con un total de 852 imágenes divididas en dos clases, 426 imágenes para la clase normal y 426 imágenes de la clase COVID-19 separadas según el nivel de gravedad de la enfermedad, es decir casos graves, moderados, leves y los etiquetados como Normal-PCR+. Esta última subclase corresponde a imágenes con discrepancia, donde el criterio del radiólogo es negativo pero la prueba RT-PCR tiene un resultado positivo al diagnóstico de la enfermedad. Por este motivo, los elementos de dicha subclase no son utilizados en este trabajo.

Para realizar los experimentos, los datos de cada clase de ambos conjuntos son divididos de forma aleatoria con una distribución de 80 % y 20 % para los conjuntos de entrenamiento y prueba, respectivamente.

4.2. Medidas de rendimiento

En este trabajo se realiza tanto clasificación binaria (para el conjunto COVIDGR) como clasificación multiclase (para el conjunto *Chest+COVID-19*); sin embargo, algunas de las medidas de rendimiento utilizadas corresponden solamente para clasificación binaria. Para poder utilizar estas medidas de rendimiento en ambos casos, la matriz de confusión se puede definir según se muestra en la Figura 4.1. En ésta, el número de casos predichos de manera correcta como no pertenecientes a una clase en específico está dado por la etiqueta Negativo Verdadero (TN, por sus siglas en inglés), el número de casos predichos de manera incorrecta como no pertenecientes a una clase en específico es representado por Negativo Falso (FN, por sus siglas en inglés), mientras que el número de casos predichos de manera errónea como pertenecientes a la clase en específico está dado por Positivo Falso (FP, por sus siglas en inglés) y aquellos predichos de manera correcta como pertenecientes a la clase especifica son

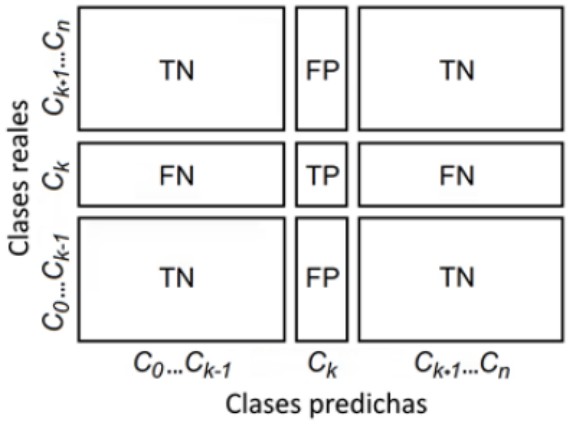


Figura 4.1: Matriz de confusión para $C > 2$ clases

denotados como Positivo Verdadero (TP, por sus siglas en inglés) (Lopez-Betancur et al., 2021).

Para evaluar el rendimiento de los modelos CNN obtenidos en los diferentes experimentos realizados en este trabajo se utilizan 5 medidas de rendimiento. Las ecuaciones para calcular las medidas de rendimiento utilizadas se muestran en el Cuadro 4.2 y sus significados son los siguientes:

- **Exactitud**: mide el porcentaje de elementos identificados correctamente en relación al total de muestras.
- **Precisión**: esta mide el número de elementos positivos identificados correctamente.
- **Sensibilidad**: esta mide la capacidad del clasificador en predecir correctamente las muestras positivas a la enfermedad, en este caso muestras positivas a neumonía por COVID-19.
- **Especificidad**: esta mide las instancias negativas clasificadas correctamente.
- **F1-score**: combina las medidas de precisión y sensibilidad en una relación que muestra la capacidad del clasificador para alcanzar mejores resultados en las clases de interés o prioritarias.
- **Índice de Exactitud Balanceada (IBA, por sus siglas en inglés)**: proporciona una medida confiable del desempeño de clasificación para problemas con conjuntos de datos desbalanceados, obteniendo valores altos cuando las exactitudes de cada clase son altas y equilibradas.

Medida de rendimiento	Ecuación
Exactitud	$TP + TN/(TP + FP + TN + FN)$
Precisión	$TP/(TP + FP)$
Sensibilidad	$TP/(TP + FN)$
Especificidad	$TN/(TN + FP)$
F1-score	$(2 * Precision * Sensibilidad)/(Precision + Sensibilidad)$
IBA	$[1 + (0.1) * (Sensibilidad - Especificidad)] * [(Sensibilidad * Especificidad)]$

Cuadro 4.2: Ecuaciones de las medidas de rendimiento

4.3. Resultados de clasificación de COVID-19 con arquitecturas CNN predefinidas

En esta sección se realizan dos experimentos de clasificación, uno para cada uno de los conjuntos de datos utilizados. En el primero, se realiza una réplica de los resultados para las tres arquitecturas CNNs con mejor rendimiento en la clasificación de COVID-19 sobre el conjunto *Chest+COVID-19*, reportados en el trabajo de Lopez-Betancur et al. (2021). En el segundo, se evalúa el rendimiento de clasificación para el conjunto COVIDGR de dos modelos residuales y una red ligera multiescala.

4.3.1. Clasificación con el conjunto *Chest+COVID-19*

Este experimento tiene como objetivo replicar los resultados obtenidos en el trabajo de Lopez-Betancur et al. (2021), con la intención de utilizar dicha información como referencia para seleccionar la arquitectura a usar para algunos de los siguientes experimentos. La réplica se realiza para las tres arquitecturas con mejor rendimiento, las cuales son: *Wide_resnet101_2*, *Resnext101_32x8d* y *Resnext50_32x4d*. En cada una de las arquitecturas, se implementa la técnica de TL, es decir, la inicialización de pesos de cada arquitectura se realiza utilizando los pesos de un respectivo modelo preentrenado. En este caso, la inicialización se realiza con los modelos preentrenados en el conjunto de datos *ImageNet* (Krizhevsky et al., 2012), los cuales son los más utilizados para TL en el ámbito de la clasificación de imágenes. Además, para realizar el entrenamiento de las arquitecturas, se utilizan los mismos hiperparámetros ajustados en Lopez-Betancur et al. (2021), como son: una LR de 0.001, 50 épocas de entrenamiento, optimizador SGD con momentum de 0.9 y un BS de 16.

En el Cuadro 4.3 se muestran los resultados de rendimiento de los tres modelos CNN utilizando el conjunto *Chest+COVID-19* aumentado. Para cada una de las arquitecturas residuales probadas, se presentan dos resultados: el primero corresponde al rendimiento de clasificación reportado en el trabajo de Lopez-Betancur et al. (2021), mientras que el segundo corresponde al rendimiento obtenido al replicar el experimento de clasificación de dicho trabajo. De manera general, los resultados de los tres modelos obtenidos con la réplica son similares, ya que se encuentran entre el 96 % y 97 % para las medidas de rendimiento utilizadas.

Arquitectura de CNN	Precisión	Sensibilidad	Especificidad	F1-score	IBA
Resnext50_32x4d (Lopez-Betancur et al., 2021)	97.75	97.75	96.07	97.75	94.04
Resnext50_32x4d	97.55	97.56	95.59	97.54	93.40
Resnext101_32x8d (Lopez-Betancur et al., 2021)	97.75	97.75	96.40	97.75	94.34
Resnext101_32x8d	97.80	97.81	96.05	97.80	94.09
Wide_resnet101_2 (Lopez-Betancur et al., 2021)	97.75	97.75	96.76	97.75	94.66
Wide_resnet101_2	97.57	97.57	96.32	97.57	94.03

Cuadro 4.3: Comparación del rendimiento de clasificación de los modelos de CNNs con los resultados reportados en (Lopez-Betancur et al., 2021)

El modelo *Resnext101_32x8d* mostró ser el que mejor se adapta al problema de clasificación, ya que supera tanto al modelo *Wide_resnet101_2* como al modelo *Resnext50_32x4d* en casi todas las medidas de rendimiento. Los resultados obtenidos son satisfactorios y consistentes a los reportados en el trabajo original, lo que demuestra la efectividad de los modelos implementados para afrontar la clasificación de COVID-19. Sin embargo, dado que el rendimiento general de las tres arquitecturas es similar, se optó por seleccionar la arquitectura *Resnext50_32x4d* para los futuros experimentos, ya que es la arquitectura más liviana de las tres, lo que se traduce en un menor tiempo de entrenamiento y un desempeño competitivo.

4.3.2. Clasificación con el conjunto COVIDGR

En este segundo experimento, se realiza la clasificación de COVID-19 sobre el conjunto de datos COVIDGR. Para obtener una mejor comparativa de los resultados, los modelos residuales utilizados corresponden a modelos de profundidad y características similares, siendo la red *ResNet50* para los modelos *resnet* y la red *resnext50_32x4d* para los modelos *resnext*; en el caso del modelo *MFL_Net* se utiliza la arquitectura de red presentada en el trabajo de Joshi and Nayak (2022). Para los modelos residuales, también se realiza una comparación de rendimiento con y sin el uso de TL. Los hiperparámetros utilizados para el entrenamiento de las redes son los mismos que en el experimento anterior, con la única diferencia de que para la red *MFL_Net* se conservó el optimizador *Adam* de la arquitectura original.

Los resultados de este experimento se muestran en el Cuadro 4.4. Al tomar en consideración únicamente los modelos sin TL, se puede observar que el rendimiento de clasificación es muy similar para los modelos *MFL_Net* y *resnext50_32x4d*, siendo superior este último tan solo por un máximo de 2 % en las medidas de rendimiento. Sin embargo, el rendimiento de clasificación de estos modelos se encuentra por debajo del 75 %. Esto denota una alta com-

Modelo de CNN	Exactitud	Precisión	Sensibilidad	Especificidad	F1-score	IBA
MFL_Net	72.07	72.19	72.25	70.85	72.07	50.74
ResNet50	69.03	70.20	69.03	70.21	69.05	48.04
resnext50_32x4d	73.55	74.16	73.54	74.18	73.60	54.41
ResNet50 TL	82.58	82.56	82.58	81.87	82.54	67.52
resnext50_32x4d TL	87.10	87.22	87.09	86.06	87.02	74.79
COVID-SDNet	81.00	84.23	76.80	–	80.07	–

Cuadro 4.4: Resultados de clasificación con el conjunto COVIDGR. El sufijo *TL* indica preentrenamiento con aprendizaje por transferencia. El símbolo '–' indica que no está disponible la información en el artículo correspondiente

plejidad para realizar la clasificación en el conjunto COVIDGR. Se puede observar que, el uso de TL claramente da un impulso al rendimiento gracias a los pesos preentrenados, otorgando un incremento significativo para los modelos residuales. Sin embargo, la red *MFL_Net*, al carecer de esta ventaja, no se puede utilizar para aumentar el rendimiento de clasificación general.

Esta comparativa muestra que el modelo *resnext50_32x4d* es el que mejor se ajusta al problema de clasificación con este conjunto. Esto se confirma al observar que, este modelo obtiene el rendimiento de clasificación más alto, tanto para los modelos sin TL como para los que sí lo implementan; en este último caso alcanzando valores alrededor del 87 % en las medidas de rendimiento.

Asimismo, se puede observar que el rendimiento de clasificación del modelo *resnext50_32x4d* con TL supera incluso al modelo *COVID-SDNet* cuyos resultados corresponden a los obtenidos en el artículo original del conjunto COVIDGR (Tabik et al., 2020). Por otra parte, el modelo *ResNet50* es el menos efectivo, ya que obtiene el menor rendimiento en general y sólo logra buenos resultados gracias a lo pesos preentrenados del TL. Con ésto, es más evidente la ventaja que ofrece el uso de TL, pues gracias a esta técnica, los modelos residuales llegan a superar incluso a los resultados del modelo propuesto para el conjunto COVIDGR en su trabajo original.

4.4. Resultados de optimización de hiperparámetros de entrenamiento

En esta sección se presentan los resultados obtenidos para la optimización de hiperparámetros de entrenamiento mediante los tres metaheurísticas utilizadas en este trabajo: DE, GA y GWO. Para cada una de ellas, se muestran los resultados obtenidos en los tres tipos de experimentos especificados en la Sección 3.2.2.

Cabe mencionar que, para llevar a cabo la experimentación correspondiente se utiliza solamente el conjunto de datos COVIDGR. Este conjunto fue seleccionado gracias a sus características, entre las que destacan: el riguroso proceso de etiquetado de las imágenes, la ausencia de marcas o etiquetas en las zona de pulmonar y la normalización del tipo de vista de la XR.

4.4.1. Optimización con DE

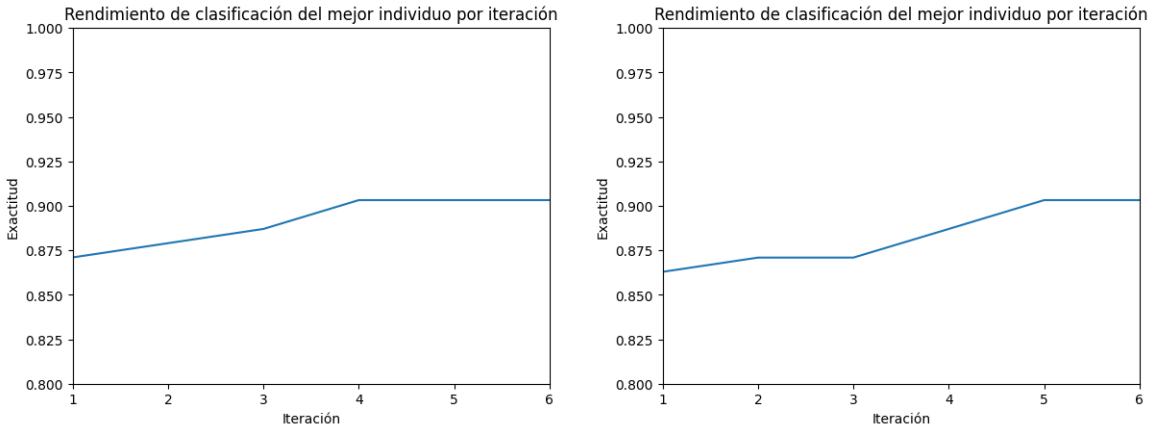
Para la optimización con DE se realizaron los tres tipos de experimentos utilizando dos variaciones de la arquitectura *resnext50_32x4d*, siendo la primera la arquitectura de red original y la segunda una variación de la misma obtenida al agregar los módulos CBAM presentados en la Sección 2.3.1. Además, en la BF se toma como referencia los resultados obtenidos en la BG para definir el nuevo espacio de búsqueda de los hiperparámetros.

Búsqueda gruesa

En la experimentación de cada una de las variaciones de la red se evaluaron aproximadamente 96 individuos diferentes. Para el experimento con la red *resnext50_32x4d* original, el tiempo total de ejecución de la optimización fue de 16 horas con 15 minutos y se alcanzó un rendimiento máximo de 90.32 % de exactitud respecto al conjunto de validación. En el caso de la optimización con *resnext50_32x4d*+CBAM, el máximo rendimiento alcanzado fue exactamente el mismo; sin embargo, este experimento resultó ser más tardado que el anterior alcanzando las 26 horas con 47 minutos. El aumento en el tiempo de ejecución se debe principalmente a la inclusión de los módulos CBAM, pues estos introducen más parámetros a la arquitectura de la red, lo que se traduce en un aumento en el tiempo de entrenamiento.

En la Figura 4.2, se muestra el comportamiento de la optimización para los dos experimentos realizados. La Figura 4.2a corresponde a la optimización con la red original y se puede observar que durante la primera iteración obtiene un individuo con un rendimiento de 87.10 % de exactitud. Durante las siguientes tres iteraciones, se presenta una mejora progresiva en el rendimiento obtenido. Esta mejora es pequeña durante la iteración 2 y 3, alcanzando un 87.90 % y 88.70 % de exactitud, respectivamente. El mayor salto de mejora se observa en la cuarta iteración donde el rendimiento sube hasta un 90.32 % de exactitud. Este resultado es el mejor obtenido durante toda la optimización, ya que durante las dos iteraciones restantes este rendimiento se mantiene.

Respecto a la optimización con *resnext50_32x4d*+CBAM (Figura 4.2b), se puede observar un comportamiento un tanto diferente. En primer lugar, el proceso de optimización inicia en un punto inferior respecto al experimento con la red original, pues durante la primera iteración se alcanza un individuo con un rendimiento de 86.29 % de exactitud. No es hasta la segunda iteración cuando se acerca al punto inicial del experimento anterior, obteniendo un rendimiento de 87.09 % de exactitud. A partir de la cuarta iteración se presenta una mejora



(a) (b)

Figura 4.2: Comportamiento del algoritmo DE durante la BG. a) Optimización con la red *resnext50_32x4d* original. b) Optimización con la red *resnext50_32x4d*+CBAM

Modelo de CNN	LR	BS	Optimizador	#Épocas	Exactitud
Resnext50_32x4d	0.004	16	SGD	20	90.32 %
Resnext50_32x4d+CBAM	0.001	8	SGD	40	90.32 %

Cuadro 4.5: Valores optimizados de los hiperparámetros de entrenamiento durante la BG usando DE y su respectiva exactitud en validación

progresiva del rendimiento obtenido llegando a un máximo de 90.32 % de exactitud en la iteración número 5, resultado que coincide con el obtenido en el experimento anterior.

Los hiperparámetros correspondientes al mejor individuo obtenido en ambos experimentos se muestran en el Cuadro 4.5 y se puede observar que, a pesar de obtener el mismo rendimiento durante la optimización, los individuos obtenidos son diferentes. Iniciando por los valores obtenidos para la LR, se observa que son muy cercanos entre sí, considerando el espacio de búsqueda del experimento, por lo que se podría considerar que la clasificación funciona bien con LRs en un rango de 10^{-3}. Para el caso del BS, no se puede decir lo mismo que con la LR pues únicamente hay tres valores para este hiperparámetro en el espacio de búsqueda. La diferencia más significativa entre ambos resultados se encuentra en el número de épocas, ya que la red *Resnext50_32x4d* original logra el mismo rendimiento que la red *Resnext50_32x4d*+CBAM pero con la mitad de épocas, aunado a un BS mayor y una LR mayor, supone un tiempo de entrenamiento mucho menor.

Modelo de CNN	LR	BS	#Épocas	Momentum	Exactitud
Resnext50_32x4d	0.003163	16	22	0.58	91.12 %
Resnext50_32x4d+CBAM	0.000415	8	37	0.4	90.32 %

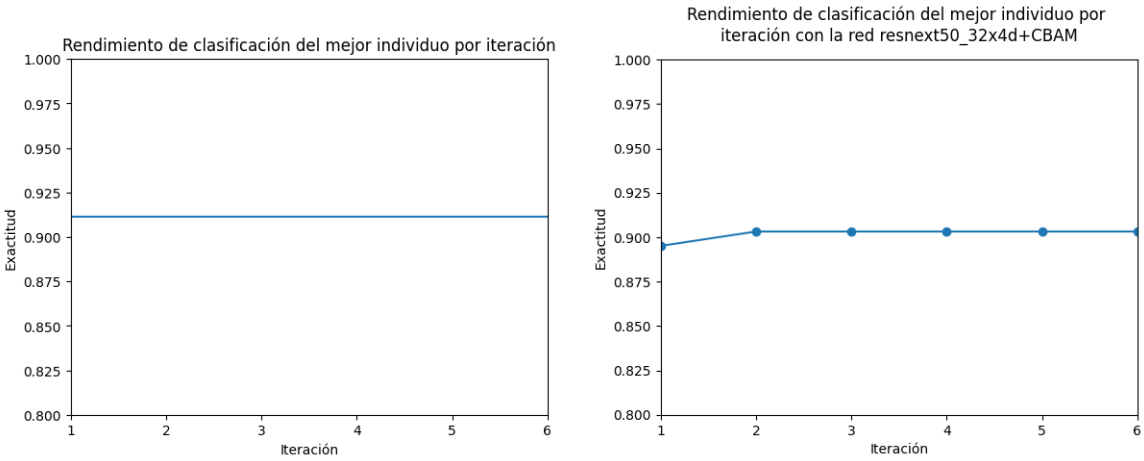
Cuadro 4.6: Valores optimizados de los hiperparámetros de entrenamiento durante la BF usando DE

Búsqueda fina

Para esta segunda búsqueda, se utilizan los resultados presentados en el Cuadro 4.5 para definir el espacio de búsqueda de los hiperparámetros a optimizar de cada una de las variaciones de red predefinidas. Al igual que el experimento anterior, se evaluaron aproximadamente 96 individuos en cada optimización. Además, los pesos de cada uno de los individuos a evaluar son inicializados utilizando el respectivo modelo ajustado durante la BG, el cual obtuvo un rendimiento del 90.32 % de exactitud en la validación para ambas redes.

El tiempo total de ejecución de la optimización con la red original fue de 22 horas con 1 minuto y se alcanzó un rendimiento máximo de 91.12 % de exactitud respecto al conjunto de validación. Los hiperparámetros correspondientes a este resultado se muestran en el Cuadro 4.6 y se observa un ajuste de los valores obteniendo un nuevo valor para la LR y el número de épocas, los cuales reflejan una ligera mejora en el rendimiento de clasificación durante la optimización. Como se observa en la Figura 4.3a, al inicializar a los individuos con los pesos obtenidos durante la BG, la optimización comienza con resultados altos, obteniendo un individuo con un rendimiento de 91.12 % de exactitud desde la primera iteración. Sin embargo, este resultado no logra ser mejorado durante el proceso de optimización, lo cual se ve reflejado en la gráfica mostrando un comportamiento constante durante toda la ejecución.

En el caso del experimento con la red *Resnext50_32x4d*+CBAM, el tiempo total de ejecución de la optimización fue de 74 horas con 19 minutos. Esto debido a que el espacio de búsqueda para el numero de épocas se realizó alrededor de las 40 épocas, además de que se utiliza un BS de 8, por lo que el tiempo de entrenamiento y validación de cada individuo es más del doble que el requerido por los individuos de la red original. El rendimiento máximo alcanzado en este experimento fue de 90.32 % de exactitud respecto al conjunto de validación. Los hiperparámetros correspondientes a este resultado se muestran en el Cuadro 4.6 y se observa un ajuste de los valores. Sin embargo, estos nuevos hiperparámetros presentan una disminución del rendimiento de clasificación respecto al resultado obtenido en la BG. Como se observa en la Figura 4.3b, al inicializar a los individuos con los pesos obtenidos durante la BG, en la primera iteración, el mejor individuo alcanza un rendimiento de clasificación de 89.51 % de exactitud y gracias a la optimización mejora a 90.32 % durante la segunda iteración, siendo ésta la máxima exactitud obtenida. El resultado obtenido, aunado al incremento en el tiempo de ejecución de la optimización, muestra un rendimiento inferior de la red *Resnext50_32x4d*+CBAM respecto a la versión original y conlleva a considerar a descartarla



(a) (b)

Figura 4.3: Comportamiento del algoritmo DE durante la BF. a) Optimización con la red *resnext50_32x4d* original. b) Optimización con la red *resnext50_32x4d*+CBAM

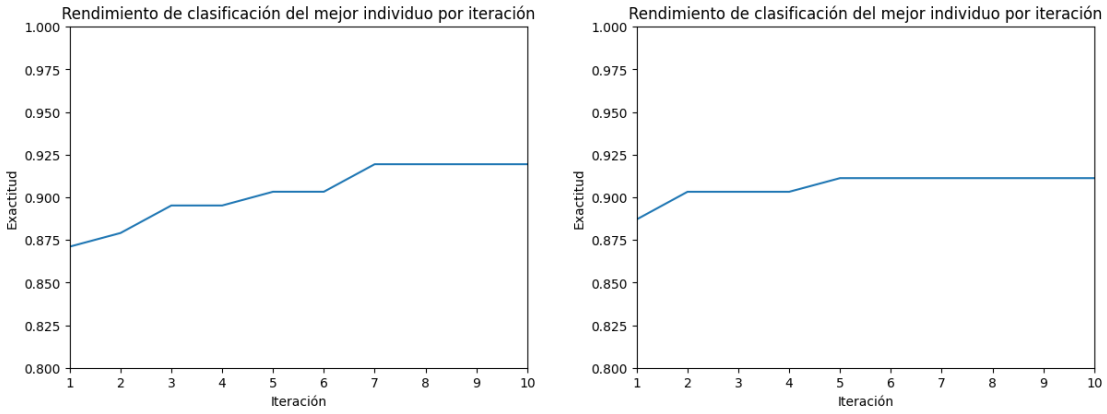
para futuros experimentos.

Búsqueda extensa

A diferencia de los experimentos anteriores, la BE es mucho más tardada. Esto se debe a que se utiliza una mayor población y el algoritmo se ejecuta durante 10 iteraciones. Durante este proceso de optimización, se evaluaron aproximadamente 330 individuos diferentes para las dos redes utilizadas (siendo más del triple de individuos que en la BG). Para el experimento con la red *resnext50_32x4d* original, el tiempo total de ejecución de la optimización fue de 66 horas con 27 minutos y se alcanzó un rendimiento máximo de 91.93 % de exactitud, respecto al conjunto de validación. En el caso de la optimización con *resnext50_32x4d*+CBAM, el máximo rendimiento alcanzado fue de 91.12 % de exactitud (el cual es el mismo que el obtenido durante la BF). Sin embargo, este experimento resultó ser más tardado, alcanzando las 90 horas con 17 minutos.

En la Figura 4.4 se muestra el comportamiento de la optimización para los dos experimentos realizados. Respecto a la Figura 4.4a, esta corresponde a la optimización con la red original y se puede observar que inicia en el mismo punto que en la BG, pues durante la primera iteración alcanza 87.10 % de exactitud. El comportamiento de este experimento de optimización refleja una mejora constante conforme avanzan las iteraciones, subiendo a un 87.90 % en la segunda iteración, 89.51 % en la tercera, 90.32 % en la quinta y alcanzando un máximo de 91.93 % en la séptima iteración. Este resultado es el mejor obtenido durante toda la optimización, ya que durante las iteraciones restantes no se obtiene alguna otra mejora.

Respecto a la optimización con *resnext50_32x4d*+CBAM mostrada en la Figura 4.4b,



(a) (b)

Figura 4.4: Comportamiento del algoritmo DE durante la BE. a) Optimización con la red *resnext50_32x4d* original. b) Optimización con la red *resnext50_32x4d*+CBAM

se puede observar un comportamiento más estático en la evolución de la optimización. El proceso de optimización inicia en un punto superior respecto al experimento con la red original, pues durante la primera iteración se alcanza un individuo con un rendimiento de 88.70 % de exactitud. Durante toda la optimización sólo se pueden observar dos mejoras del rendimiento. La primera se da en la iteración número dos, en la cual el rendimiento del mejor individuo alcanza una exactitud del 90.32 %. La última mejora se observa en la quinta iteración, donde la optimización alcanza su máximo de 91.12 % de exactitud, resultado que coincide con el obtenido durante la BF.

Los hiperparámetros correspondientes al mejor individuo obtenido en ambos experimentos se muestran en el Cuadro 4.7. Se pueden observar ciertas similitudes con los experimentos de la BG. En el caso del BS, los resultados coinciden de manera exacta, con un BS de 16 para la red original y de 8 para la implementación con los bloques CBAM. Un dato a destacar es que en ninguno de los experimentos realizados hasta este punto se ha obtenido un valor de BS de 32, por lo que este valor no parece ajustarse bien a la tarea de clasificación. Para la LR, se observa que los valores obtenidos se siguen manteniendo en el rango de 10^{-3} o en caso de la *Resnext50_32x4d*+CBAM se obtiene el valor más cercano a este rango, según el espacio de búsqueda. Finalmente, el número de épocas no dista tanto como en la BG y se mantiene en promedio de 20 épocas de entrenamiento.

Tomando en consideración los resultados obtenidos en la optimización con DE, se pudo comprobar que la red original presenta un mejor comportamiento que la versión con los módulos CBAM. En los tres experimento realizados, la red *Resnext50_32x4d*+ CBAM mostró una mayor dificultad para igualar el rendimiento de la red original, siendo generalmente inferior aún con un tiempo de optimización mucho más elevado; ésto se puede observar principalmente en la BF y BE. Por lo que, retomando lo mencionado en la BF, se decidió no

Modelo de CNN	LR	BS	Optimizador	#Épocas	Exactitud
Resnext50_32x4d	0.007	16	SGD	17	91.93 %
Resnext50_32x4d+CBAM	0.0007	8	SGD	23	91.12 %

Cuadro 4.7: Valores optimizados de los hiperparámetros de entrenamiento durante la BE usando DE

Modelo de CNN	LR	BS	Optimizador	#Épocas	Exactitud
Resnext50_32x4d	0.0007	8	SGD	22	89.51 %

Cuadro 4.8: Valores optimizados de los hiperparámetros de entrenamiento durante la BG utilizando GA

utilizar la versión *resnext50_32x4d*+CBAM para la optimización con GA y GWO.

4.4.2. Optimización con GA

Para la optimización con GA, se realizan los tres tipos de experimentos (BG, BF y BE), utilizando como arquitectura predefinida únicamente a la red *Resnext50_32x4d*. Los parámetros utilizados para el algoritmo GA se detallan en la Sección 3.2.2 a excepción del número de iteraciones y tamaño de la población, los cuales dependen del tipo de experimento a realizar.

Búsqueda gruesa

Durante el experimento de la BG, el tiempo total de ejecución de la optimización fue de 7 horas con 7 minutos y se alcanzó un rendimiento máximo de 89.51 % de exactitud, respecto al conjunto de validación. La Figura 4.5 muestra el comportamiento de la optimización para cada una de las iteraciones del algoritmo. La optimización de hiperparámetros inicia obteniendo un rendimiento máximo de 87.09 % de exactitud durante la primera iteración. En la iteración 2 y 3, se observa una mejora progresiva del rendimiento, obteniendo un 88.70 % durante la segunda iteración y llegando a un máximo de 89.51 % de exactitud en la tercera iteración.

Los hiperparámetros correspondientes a este resultado se muestran en el Cuadro 4.8, donde se pueden observar valores muy parecidos a los obtenidos por la red *Resnext50_32x4d*+CBAM durante la BE con DE, ya que coinciden tanto la LR como el BS y tan sólo se presenta una mínima diferencia en el número de épocas. Sin embargo, el rendimiento del modelo es ligeramente inferior, siendo el rendimiento más bajo obtenido entre los experimentos realizados.

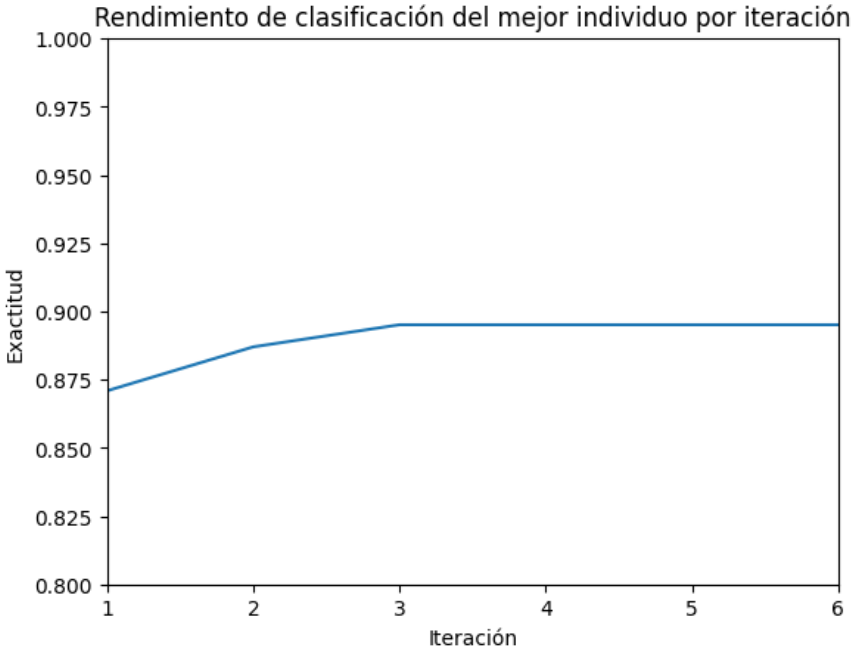


Figura 4.5: Comportamiento del algoritmo GA durante la BG

Búsqueda fina

En la BF, el espacio de búsqueda de los hiperparámetros de entrenamiento a optimizar es definido tomando como referencia los valores del mejor individuo obtenido durante la BG. Los límites, tanto de la LR como el número de épocas, se ubican alrededor de dicho resultado utilizando un margen de ± 0.0001 para la LR y ± 5 para el número de épocas. En el caso del BS y el optimizador se fijan en 8 y SGD, respectivamente. Además, se busca el parámetro de momentum para el optimizador SGD en un rango de 0 a 1.

La Figura 4.6 muestra el comportamiento de la optimización para cada una de las iteraciones del algoritmo. La optimización inicia obteniendo un rendimiento superior al presentado durante la BG, alcanzando una exactitud de 90.32 % durante la primera iteración. Sin embargo, se puede observar un comportamiento estático ya que solo se presenta una mejora durante la optimización, la cual corresponde al salto mostrado en la gráfica durante la segunda iteración. De esta manera se alcanza un 91.12 % de exactitud, siendo este el mejor rendimiento obtenido respecto al conjunto de validación con un tiempo total de ejecución de 6 horas con 25 minutos.

Los hiperparámetros correspondientes a este resultado se muestran en el Cuadro 4.9, donde se pueden observar el ajuste de los valores respecto a los presentados en la BG. Los nuevos valores obtenidos resultan ser muy similares a los anteriores. A pesar de que la LR tan sólo redujo su valor en 0.00003 y el número de épocas no presenta cambio alguno, se

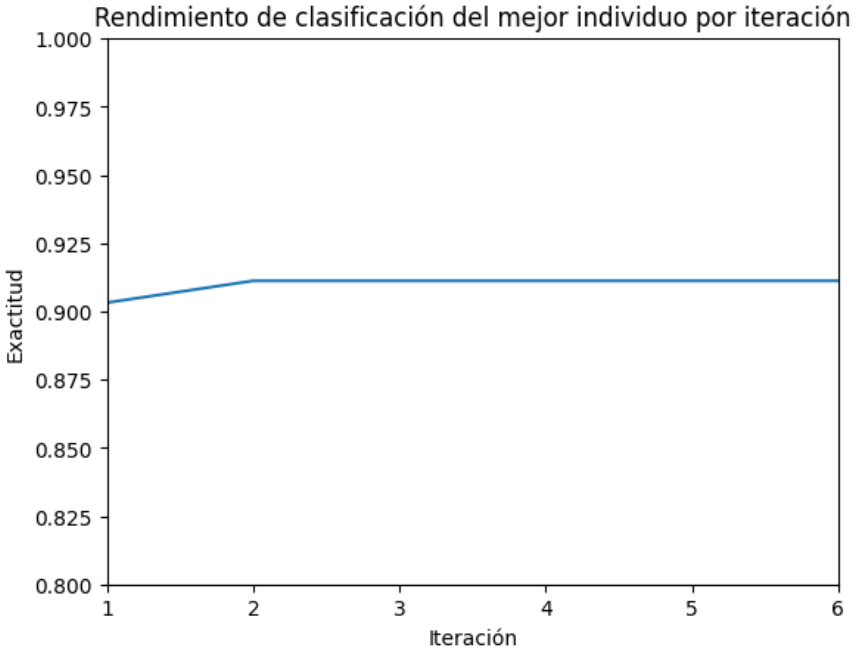


Figura 4.6: Comportamiento del algoritmo GA durante la BF

Modelo de CNN	LR	BS	#Épocas	Momentum	Exactitud
Resnext50_32x4d	0.00066	8	22	0.26	91.12 %

Cuadro 4.9: Valores optimizados de los hiperparámetros de entrenamiento durante la BF usando GA

obtuvo una mejora considerable en el rendimiento de clasificación, lo cual se puede deber principalmente al ajuste de momentum para el optimizador SGD.

Búsqueda extensa

Durante el proceso de optimización con la BE, se evaluaron aproximadamente 330 individuos. El tiempo total de ejecución de la optimización fue de 20 horas con 59 minutos y se alcanzó un rendimiento máximo de 91.12 % de exactitud, respecto al conjunto de validación.

En la Figura 4.7 se muestra el comportamiento de la optimización para la BE. En la gráfica se puede observar una mejora progresiva del rendimiento de clasificación marcada por tres puntos. En un inicio, el mejor individuo encontrado alcanza un rendimiento de clasificación de 88.70 % en la exactitud, por lo que el aumento de la diversidad de información en la metaheurística logra mejorar el punto de partida respecto a lo obtenido en la BG. Durante la segunda iteración, se observa un salto significativo de mejora en el rendimiento,

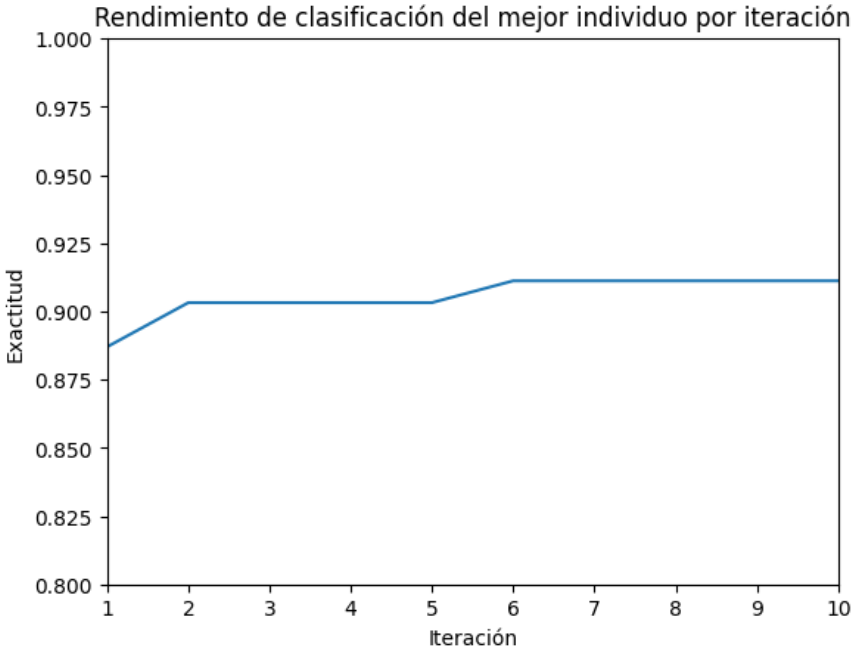


Figura 4.7: Comportamiento del algoritmo GA durante la BE

Modelo de CNN	LR	BS	Optimizador	#Épocas	Exactitud
Resnext50_32x4d	0.001	8	SGD	22	91.12 %

Cuadro 4.10: Valores optimizados de los hiperparámetros de entrenamiento durante la BE usando GA

alcanzando un 90.32 % de exactitud y superando desde este punto al mejor resultado obtenido por la BG. Finalmente, durante la sexta iteración se obtiene el máximo rendimiento de clasificación obtenido en el experimento, logrando igualar al resultado obtenido con la BF con un 91.12 % de exactitud.

En el Cuadro 4.10, se muestran los valores de los hiperparámetros encontrados durante la BE. Se puede observar que, estos valores son muy similares a los obtenidos en los experimentos de BG y BF. Principalmente con el número de épocas y BS, para los cuales se obtienen los mismos valores durante los tres experimentos, siendo 22 y 8, respectivamente. En el caso de la LR, se muestran variaciones según el experimento; no obstante, los resultados obtenidos son muy cercanos tomando en consideración los valores posibles del espacio de búsqueda. De igual manera, el máximo rendimiento de clasificación obtenido por los experimentos coincide tanto para la BF como para la BE. Sin embargo, se puede decir que el rendimiento de la optimización con GA es inferior en comparación con el obtenido por la DE.

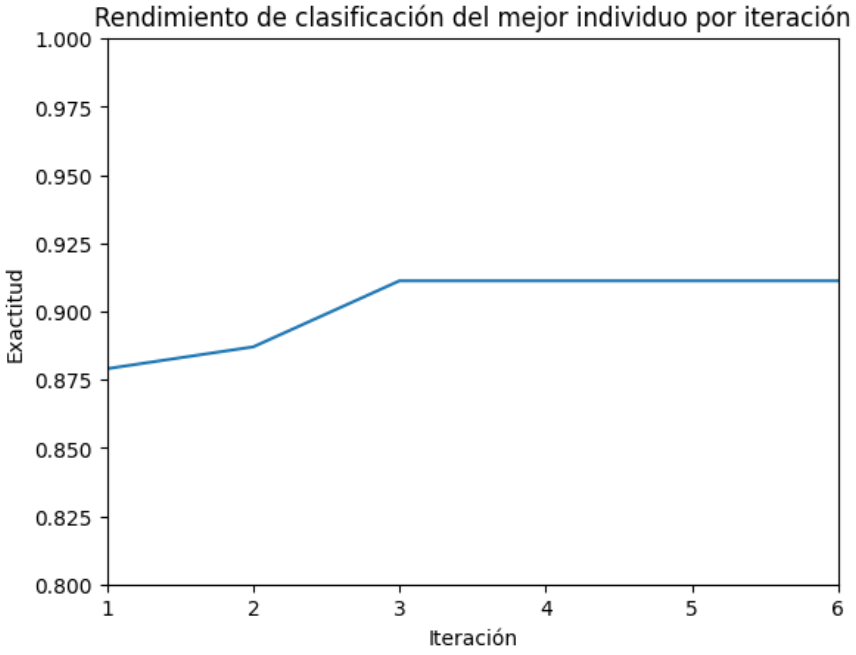


Figura 4.8: Comportamiento del algoritmo GWO durante la BG

4.4.3. Optimización con GWO

Para la optimización con GWO, se realizan los tres tipos de experimentos (BG, BF y BE), utilizando como arquitectura predefinida únicamente a la red *Resnext50_32x4d*. Los parámetros utilizados para el algoritmo GWO se detallan en la Sección 3.2.2 a excepción del número de iteraciones y tamaño de la población, los cuales dependen del tipo de experimento a realizar.

Búsqueda gruesa

El tiempo total de ejecución de la optimización fue de 16 horas con 32 minutos y se alcanzó un rendimiento máximo de 91.12 % de exactitud respecto al conjunto de validación. Tanto el tiempo de ejecución como el rendimiento de clasificación son muy similares a lo obtenido en la BG con DE. En la Figura 4.8 se muestra el comportamiento de la optimización del experimento con la BG. El proceso de optimización inicia en un punto elevado, en comparación a experimentos anteriores, alcanzando un 88.70 % desde la primera iteración. Durante el proceso de optimización, únicamente se presentan dos mejoras en el rendimiento, la primera en la iteración 2, donde se obtiene un 89.51 de exactitud y la segunda en la iteración 5, donde se alcanza el máximo rendimiento de 91.12 %.

Los hiperparámetros correspondientes al mejor individuo obtenido se muestran en el Cuadro 4.11. El valor obtenido para el BS es de 16, lo cual es la principal diferencia con los

Modelo de CNN	LR	BS	Optimizador	#Épocas	Exactitud
Resnext50_32x4d	0.001	16	SGD	22	91.12 %

Cuadro 4.11: Valores optimizados de los hiperparámetros de entrenamiento durante la BG usando GWO

Modelo de CNN	LR	BS	#Épocas	Momentum	Exactitud
Resnext50_32x4d	0.000348	16	25	0.39	91.93 %

Cuadro 4.12: Valores optimizados de los hiperparámetros de entrenamiento durante la BF usando GWO

experimentos de GA, pero es consistente con lo obtenido en los resultados de la DE. Para el número de épocas, nuevamente vemos un valor de 22 coincidiendo con los resultados de GA y no distando mucho de lo obtenido por la DE. De igual manera, la LR se mantiene en un rango de 10^{-3}.

Búsqueda fina

En el experimento de BF, el espacio de búsqueda de los hiperparámetros de entrenamiento a optimizar es definido tomando como referencia los valores de los hiperparámetros obtenidos durante la BG. Los límites, tanto de la LR como el número de épocas, se ubican alrededor de dicho resultado utilizando un margen de ± 0.001 para la LR y ± 5 para el número de épocas. En el caso del BS se fija en 16. Además, se busca el parámetro de momentum para el optimizador SGD en un rango de 0 a 1.

La Figura 4.9 muestra el comportamiento de la optimización para cada una de las iteraciones del algoritmo. El tiempo total de ejecución fue de 16 horas con 15 minutos y al igual que en los otros experimentos de BF, la optimización inicia obteniendo un rendimiento relativamente alto, siendo muy cercano al obtenido durante la BG pues se alcanza una exactitud de 90.32 %. Con esto se puede observar que, los pesos obtenidos durante la BG aportan un buen impulso al proceso de clasificación. De esta manera se alcanza el rendimiento máximo de 91.93 % de exactitud durante la segunda iteración, logrando alcanzar a lo obtenido durante la BE con la DE y siendo estos dos experimentos los que mejor rendimiento presentan en la optimización de hiperparámetros de entrenamiento.

Los hiperparámetros correspondientes al individuo encontrado se presentan en el Cuadro 4.12, donde se observa una considerable disminución en la LR respecto a la BG. Asimismo, se observa un ajuste al número de épocas de entrenamiento, el cual incrementa a 25. En este experimento de BF, se obtuvo un ajuste de los tres hiperparámetros a optimizar, lo cual da como resultado un incremento del rendimiento de clasificación respecto a la BG.

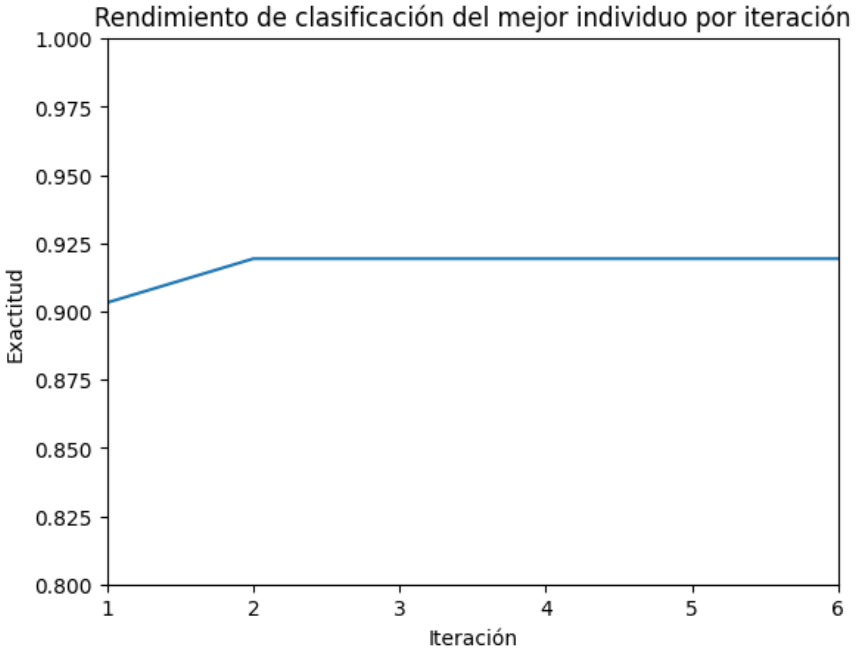


Figura 4.9: Comportamiento del algoritmo GWO durante la BF

Búsqueda extensa

Durante el experimento con la BE, el tiempo total de ejecución de la optimización fue de 53 horas con 47 minutos y se alcanzó un rendimiento máximo de 91.12 % de exactitud, respecto al conjunto de validación. La gráfica de la Figura 4.10 muestra el comportamiento de la optimización para la BE y se puede observar una mejora progresiva del rendimiento de clasificación. Los saltos de mejora presentados se observan principalmente durante las primeras iteraciones, iniciando con un rendimiento de clasificación de 87.90 % en la exactitud. Este rendimiento va mejorando durante la iteración 2 y 3 alcanzando un 89.51 % y 90.32 % de exactitud, respectivamente. Finalmente, durante la sexta iteración se logra alcanzar el rendimiento de clasificación máximo obtenido en este experimento. En general, el comportamiento de la optimización muestra una mejora progresiva del rendimiento de clasificación, pero el resultado final no es mejor al de los experimentos anteriores, tan solo logra igualar a lo obtenido durante la BG.

En el Cuadro 4.13 se presentan los valores de los hiperparámetros encontrados durante la BE. Se puede observar que estos valores son muy similares a los obtenidos durante la BG, un número de épocas muy cercano y la misma LR. La principal diferencia se puede encontrar en el BS, el cual obtuvo un valor de 8, siendo un valor diferente a lo obtenido en los demás experimentos con GWO y DE pero muy recurrente durante la optimización con GA.

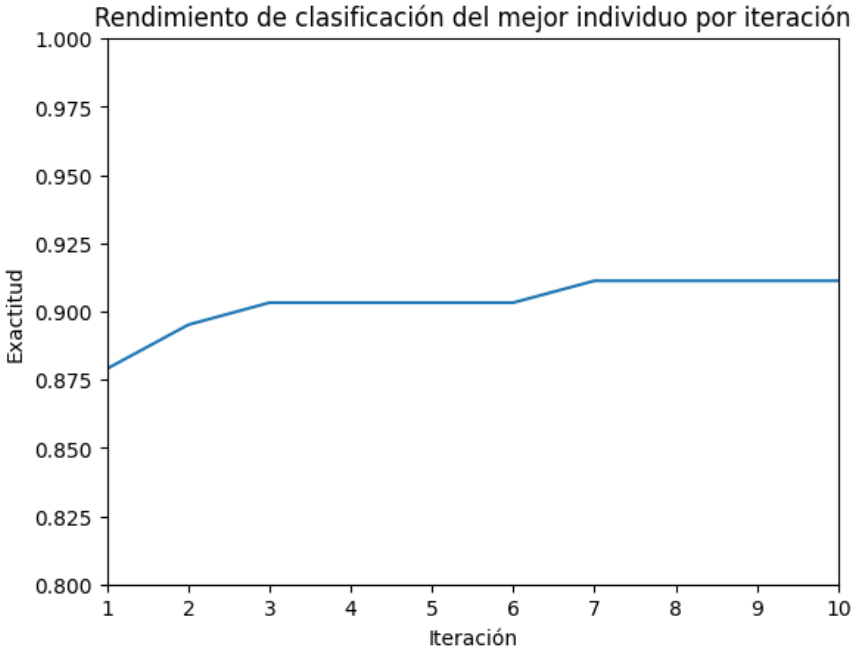


Figura 4.10: Comportamiento del algoritmo GWO durante la BE

Modelo de CNN	LR	BS	Optimizador	#Épocas	Exactitud
Resnext50_32x4d	0.001	8	SGD	21	91.12 %

Cuadro 4.13: Valores optimizados de los hiperparámetros de entrenamiento durante la BE usando GWO

4.4.4. Validación de resultados

Para obtener una evaluación más real de los resultados mostrados, se utilizaron los dos mejores modelos obtenidos mediante la optimización para evaluar el rendimiento de clasificación respecto al conjunto de prueba. Los dos modelos seleccionados corresponden al obtenido mediante el experimento de BE con DE y el obtenido en la BF utilizando GWO, ambos modelos obtuvieron un rendimiento de clasificación de 91.93 % de exactitud respecto al conjunto de validación.

Para realizar dicho experimento, se fusionó el conjunto de entrenamiento y validación, utilizado durante el proceso de optimización, para crear el nuevo conjunto de entrenamiento. La red *Resnext50_32x4d* se entrenó utilizando los respectivos valores de hiperparámetros encontrados en la optimización y para la inicialización de pesos se utilizó el archivo correspondiente a cada uno de los modelos probados.

En el Cuadro 4.14 se muestra el rendimiento de clasificación obtenido para cada modelo y

Modelo de optimización	Exactitud	Precisión	Sensibilidad	Especificidad	F1-score	IBA
Resnext50_32x4d+DE_BE	90.32	90.34	90.32	90.26	90.32	81.53
Resnext50_32x4d+GWO_BF	89.67	89.67	89.67	89.48	89.67	80.25
COVID-SDNet	81.00	84.23	76.80	–	80.07	–

Cuadro 4.14: Rendimiento de clasificación respecto al conjunto de prueba de los mejores modelos con hiperparámetros de entrenamiento optimizados

se compara con el rendimiento obtenido por el modelo *COVID-SDNet*, originario del trabajo donde se presentó el conjunto de datos COVIDGR. En ambos modelos obtenidos mediante este trabajo, se puede observar una ligera disminución de rendimiento de clasificación respecto al conjunto de entrenamiento. Este comportamiento es de esperarse, ya que el conjunto de prueba contiene únicamente datos que no se utilizaron durante el proceso de optimización, por lo tanto, son desconocidos para los modelos utilizados. Por otra parte, ambos modelos obtenidos mediante la optimización logran superar el rendimiento de clasificación obtenido por la red *COVID-SDNet*, mostrando una mejor adaptación al problema por parte de la optimización de los hiperparámetros. Aun cuando ambos obtuvieron un rendimiento similar durante la optimización no sucede lo mismo durante la evaluación con el conjunto de prueba, ya que el modelo *Resnext50_32x4d*+GWO_BF no logró alcanzar en exactitud al modelo *Resnext50_32x4d*+DE_BE. Entonces, este último modelo es el que mejor se adapta a la tarea de clasificación de COVID-19 para el conjunto COVIDGR.

Durante la ejecución de los experimentos se observó que, el comportamiento de los mismos puede presentar cierta variación en los resultados. Es por ello que, se decidió realizar una validación cruzada (k_fold CV, por sus siglas en inglés), con el objetivo de obtener una mejor estimación del comportamiento del modelo *Resnext50_32x4d*+DE_BE. Para esto, se utilizó una configuración 5_fold sobre el conjunto de entrenamiento y validación para crear los subconjuntos utilizados para entrenar y evaluar dicho modelo. Los resultados de la CV se muestran en el Cuadro 4.15 y se logra observar la variación del rendimiento de clasificación según la distribución de los datos. La partición 1 y 5 obtienen el mismo resultado, siendo las que menor rendimiento de clasificación alcanzan. De manera similar, la partición 2 y 3 también comparten resultado, logrando superar a lo obtenido en los resultados con el particionamiento utilizado durante los experimentos de optimización. En el caso de la partición 4 presenta un rendimiento de clasificación del 100 % de exactitud, superando de manera significativa a los demás resultados de la validación y es principalmente debido a esta partición que se que se obtiene un valor cercano a 4 % en la desviación estándar (Std). El incremento visto en el rendimiento de clasificación es causado gracias a la inicialización de pesos del modelo, el cual se realiza utilizando el archivo de pesos del modelo obtenido durante la optimización. De esta manera, las diferentes particiones aleatorias de la CV contiene datos ya conocidos por el modelo a evaluar.

Como último experimento, se repitió la optimización de hiperparámetros de entrenamien-

Partición	Exactitud
fold_1	89.60
fold_2	95.16
fold_3	95.16
fold_4	100
fold_5	89.60
Media	93.90
Std	3.93

Cuadro 4.15: Rendimiento de clasificación obtenido mediante la CV con configuración 5_fold para el modelo *Resnext50_32x4d*+DE_BE

to utilizando la misma configuración con la que se obtuvo el mejor modelo, esto es, utilizando la BE del algoritmo DE. Este proceso se realizó cinco veces, con el objetivo de observar la variación en los resultados obtenidos. En el Cuadro 4.16 se muestran los resultados obtenidos, incluyendo el mejor resultado del Cuadro 4.14. Se puede observar que, la LR no varía mucho respecto al espacio de búsqueda, manteniéndose en el rango de 10^{-3}. En los resultados para el BS incluyen los tres valores posibles pero se centra en 8 y 16, por lo que considerando que se está utilizando un conjunto de datos no muy grande es normal que los valores de BS más pequeños den mejores resultados. En el número de épocas es donde se puede ver la mayor variación de los resultados, sin embargo se puede observar que se centran en dos puntos, alrededor de 20 y 45 épocas, lo cual es consistente con los resultados obtenidos incluso con las otras metaheurísticas y sus respectivos experimentos realizados. Respecto al rendimiento de clasificación, la variación de los resultados obtenidos con el conjunto de validación (CVal) no es tan significativa, obteniendo una desviación estándar de 0.5, la cual, aumenta al evaluar el modelo con el conjunto de prueba (CTest). Sin embargo, tanto la variación como la disminución en el rendimiento son aceptables, tomando en cuenta que el conjunto de prueba contiene datos que no fueron utilizados durante la optimización y son desconocidos por los modelos obtenidos.

4.5. Resultados de optimización de hiperparámetros de estructura

En esta sección se presentan los resultados obtenidos de la optimización de hiperparámetros de estructura. Durante los experimentos mostrados en la sección anterior se observa que el algoritmo DE presenta los mejores resultados, por lo que en esta sección solo se presentan los resultados de optimización utilizando únicamente este algoritmo. La optimización se aplica a las tres propuestas para la generación de la estructura de la red: modelo convolucional

Experimento	LR	BS	Optimizador	#Épocas	Exactitud (CVal)	Exactitud (CTest)
DEBE_Org	0.007	16	SGD	17	91.93 %	90.32 %
DE_BE_1	0.004	8	SGD	46	91.12 %	87.74 %
DE_BE_2	0.001	8	SGD	44	91.93 %	89.67 %
DE_BE_3	0.001	16	SGD	22	91.12 %	86.45 %
DE_BE_4	0.007	32	SGD	46	91.93 %	88.56 %
DE_BE_5	0.004	8	SGD	36	92.74 %	89.67 %
Promedio					91.79	88.73 %
Std					0.5566	1.323

Cuadro 4.16: Variabilidad de resultados para optimización de hiperparámetros de entrenamiento con la configuración DE+BE

básico, modelo multiescala y modelo residual. Para estos experimentos, se utiliza el conjunto de datos COVIDGR y los detalles sobre los parámetros y la implementación del algoritmo DE se describen en la Sección 3.2.2.

4.5.1. Optimización del modelo convolucional básico

El enfoque del modelo convolucional básico consiste en la optimización de una estructura de red, tanto para el apartado de extracción de características con los bloques convolucionales básicos como para el apartado de clasificación con las capas densas. La optimización de los hiperparámetros se realiza utilizando los tres tipos de experimentos definidos en éste documento (BG, BF y BE). Las especificaciones de los experimentos mencionados y del enfoque convolucional básico se presentan en la Sección 3.2.2.

Búsqueda gruesa

Para el experimento de BG del modelo convolucional básico, la ejecución del proceso de optimización se llevó a cabo durante 6 iteraciones, con una población de 15 individuos para cada una. La duración total del experimento fue de 21 horas con 51 minutos, resultando en un tiempo de ejecución muy similar a las BGs de la optimización de hiperparámetros de entrenamiento. Las estructuras generadas siguen el enfoque convolucional básico, por lo que resultan ser significativamente menos profundas en comparación a la red *Resnext50_32x4d*, llegando a tener un máximo de tan solo 6 capas convolucionales. Sin embargo, el uso de una capa *Flatten()* para el aplanado de los mapas de características, en conjunto con las capas densas, pueden aumentar ligeramente el tiempo de entrenamiento, siendo esta la principal razón de los tiempos de ejecución tan similares entre la optimización del enfoque convolucional básico y una red profunda como la *Resnext50_32x4d*.

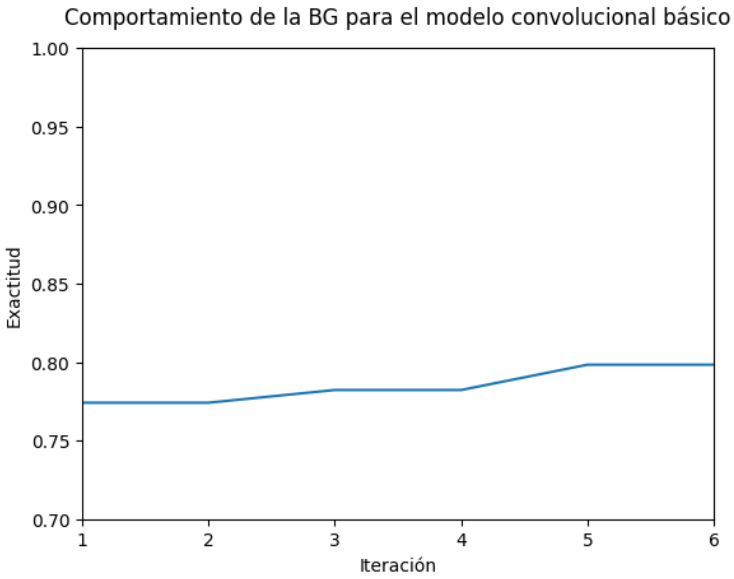


Figura 4.11: Comportamiento de la optimización del modelo convolucional básico con el algoritmo DE durante la BG

En la Figura 4.11 se muestra el comportamiento de la optimización con la BG. Se puede observar que, una estructura conformada únicamente por capas convolucionales básicas y capas densas presenta dificultades para afrontar la clasificación de COVID-19. Esto se puede notar observando los resultados obtenidos, pues el proceso de optimización inicia en un valor inferior a experimentos anteriores, alcanzando una exactitud de 77.41 % durante las dos primeras iteraciones. Durante la iteración 3 y 4 se logra una ligera mejora de rendimiento, llegando a obtener un 78.22 % de exactitud. Finalmente, el máximo rendimiento obtenido con esta configuración experimental es de 79.83 % de exactitud, respecto al conjunto de validación, alcanzado en la iteración 5. A pesar de obtener un rendimiento inferior, respecto a los resultados obtenidos durante la optimización de hiperparámetros de entrenamiento, hay que recordar que este resultado se logra con una arquitectura de red menos profunda, más simple y sin el impulso que representa el uso de TL con *ImageNet* utilizado en la red *Resnext50_32x4d*.

En el Cuadro 4.17 se muestra el modelo óptimo encontrado por la BG del algoritmo DE. La estructura de la red optimizada recibe como entrada imágenes con dimensión de 224×224. La sección de extracción de características se compone de 4 bloques convolucionales con 167, 351, 61 y 194 filtros de tamaño 3×3 respectivamente, cada uno de estos bloques consta de una capa *Conv2D()* con una activación $ReLU$ y su respectivo número y tamaño de filtro; además de una capa *MaxPool2D*(2×2). La sección de clasificación de la red inicia con un aplanado de los mapas de características por medio de una capa *Flatten()*, seguido de tres capas *Dense()* con activación $ReLU$ y su correspondiente número de neuronas (109, 110

#Épocas	147	
BS	23	
Exactitud	79.83 %	
Nombre de Capa	Descripción	Salida
BC_1	Conv(167, 3×3) MaxPool(2×2)	112×112
BC_2	Conv(351, 3×3) MaxPool(2×2)	56×56
BC_3	Conv(61, 3×3) MaxPool(2×2)	28×28
BC_4	Conv(194, 3×3) MaxPool(2×2)	14×14
Flatten		1×38024
Dense_1	Dense(109, ReLU)	1×109
Dense_2	Dense(110, ReLU)	1×110
Dense_3	Dense(197, ReLU)	1×197
Out	Dense(2, softmax)	1×2

Cuadro 4.17: Estructura optimizada del enfoque convolucional básico durante la BG

y 197). Finalmente, la capa de salida es una capa *Dense()* de dos neuronas y activación *softmax*.

En cuanto al entrenamiento, el modelo fue entrenado por 147 épocas con un BS de 23, utilizando el optimizador SGD con una LR de 0.001 y momentum de 0.9. Con esta configuración se obtiene una exactitud de 79.83 % respecto al conjunto de validación.

Búsqueda fina

Para este experimento con el enfoque convolucional básico se utiliza la configuración establecida para la BF. Esto es, se utilizan 6 iteraciones para la optimización con una población de 15 individuos en cada una. Aunado a esto, la búsqueda de los hiperparámetros se realiza alrededor de los resultados obtenidos durante la BG, utilizando: ± 1 tanto para el número de bloques convolucionales como el número de capas densas, ± 50 para la cantidad de filtros/neuronas en las capas, ± 5 para el BS y ± 10 en el número de épocas de entrenamiento. Una de las principales diferencias con los experimentos de BF realizados en la sección 4.4 es que, al modificar la estructura de la red no es posible utilizar el modelo obtenido durante la BG para impulsar el rendimiento de clasificación. Por lo que, para este experimento la optimización inicia desde cero.

El tiempo total de ejecución fue de 34 horas con 35 minutos y al iniciar con la optimización desde cero, el rendimiento de clasificación inicia en un punto similar al de la BG, alcanzando una exactitud de 77.64 % durante las dos primeras iteraciones. El máximo rendimiento de clasificación obtenido durante este experimento se encuentra en la tercera iteración, co-

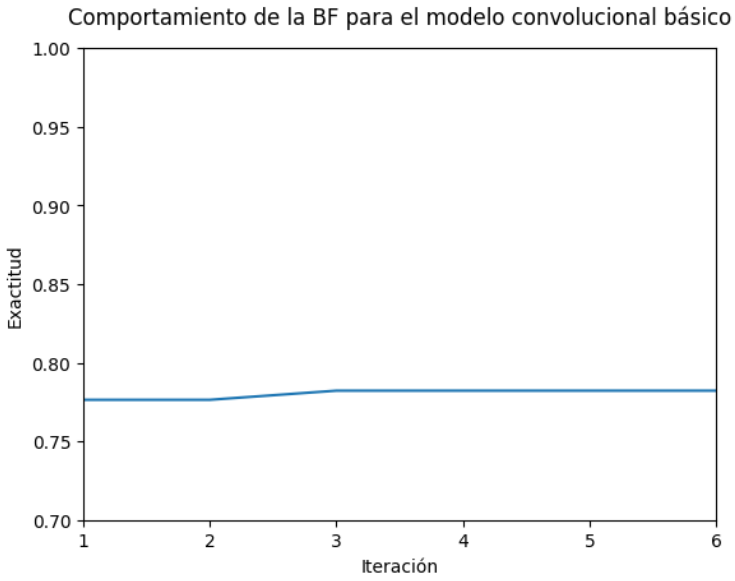


Figura 4.12: Comportamiento de la optimización del modelo convolucional básico con el algoritmo DE durante la BF

rrespondiendo a un 78.22 % de exactitud respecto al conjunto de validación. La Figura 4.12 muestra estos resultados y se observa un comportamiento casi constante donde el resultado final no logra superar a lo obtenido durante la BG, por lo que para este experimento la BF no logra buenos resultados.

El modelo óptimo encontrado por la BF del algoritmo DE se muestra en el Cuadro 4.18. La estructura de la red optimizada presenta múltiples diferencias a lo obtenido en el experimento anterior. En esta ocasión, la sección de extracción de características se compone de 3 bloques convolucionales con 167, 312 y 76 filtros de tamaño 3×3 respectivamente y con una entrada para imágenes con dimensión 224×224. Los bloques convolucionales mantienen la estructura presentada anteriormente, es decir, una capa *Conv2D()* con una activación *ReLU* y una capa *MaxPool2D*(2×2). La sección de clasificación de la red ahora cuenta con cuatro capas *Dense()* con activación *ReLU* y su correspondiente número de neuronas (225, 112, 149 y 246). Finalmente, la capa de salida es una capa *Dense()* de dos neuronas y activación *softmax*.

En cuanto al entrenamiento, el modelo fue entrenado por 112 épocas con un BS de 22 y utilizando el optimizador SGD con una LR de 0.001 y momentum de 0.9;. De esta manera, se obtiene una exactitud de 78.22 % respecto al conjunto de validación. En general, este modelo cuenta con un bloque convolucional menos y una capa densa más que el modelo obtenido por la BG, el tamaño de lote es muy similar y la principal diferencia se encuentra en las épocas de entrenamiento y el número de neuronas o filtros obtenidos. El hecho de tener un bloque convolucional menos resulta en mapas de características de mayor tamaño (28×28),

#Épocas	112	
BS	22	
Exactitud	78.22 %	
Nombre de Capa	Descripción	Salida
BC_1	Conv(167, 3×3) MaxPool(2×2)	112×112
BC_2	Conv(312, 3×3) MaxPool(2×2)	56×56
BC_3	Conv(76, 3×3) MaxPool(2×2)	28×28
Flatten		1×59584
Dense_1	Dense(225, ReLU)	1×225
Dense_2	Dense(112, ReLU)	1×112
Dense_3	Dense(149, ReLU)	1×149
Dense_4	Dense(246, ReLU)	1×246
Out	Dense(2, softmax)	1×2

Cuadro 4.18: Estructura optimizada del enfoque convolucional básico durante la BF

los cuales, al realizar el aplanado, da como resultado un vector de casi el doble del tamaño que el obtenido en el modelo de la BG. Aunado a esto, el uso de una capa densa extra resulta en un incremento significativo de los parámetros de la estructura, el cual no es adecuado, ya que el rendimiento de clasificación se ve reducido.

Búsqueda extensa

La ejecución del proceso de optimización del experimento de BE con el modelo convolucional básico, se llevó a cabo durante 10 iteraciones con una población de 30 individuos para cada una. La duración total del experimento fue de 95 horas con 51 minutos, resultando en un tiempo de ejecución superior al de cualquiera de los experimentos de optimización de hiperparámetros de entrenamiento. El incremento en el tiempo de ejecución se debe a aquellos individuos con los que se generan estructuras con una gran cantidad de parámetros (con un vector de aplanado significativamente grande o aquellos con muchas capas densas), lo que provoca un incremento en el tiempo de entrenamiento que supera al de una red mucho más profunda como la red *Resnext50_32x4d* utilizada en los experimentos de optimización de hiperparámetros de entrenamiento, independientemente de las épocas destinadas para entrenar el modelo.

En la Figura 4.13 se muestra que, la optimización con la BE tiene el mejor comportamiento de los experimentos con el enfoque convolucional básico. Se puede observar que, el proceso de optimización inicia obteniendo un rendimiento similar al mejor resultado de la BG con una exactitud de 79.83 %. Dicho rendimiento de clasificación sigue mejorando durante las tres iteraciones siguientes, alcanzando un 80.64 % de exactitud durante la segunda

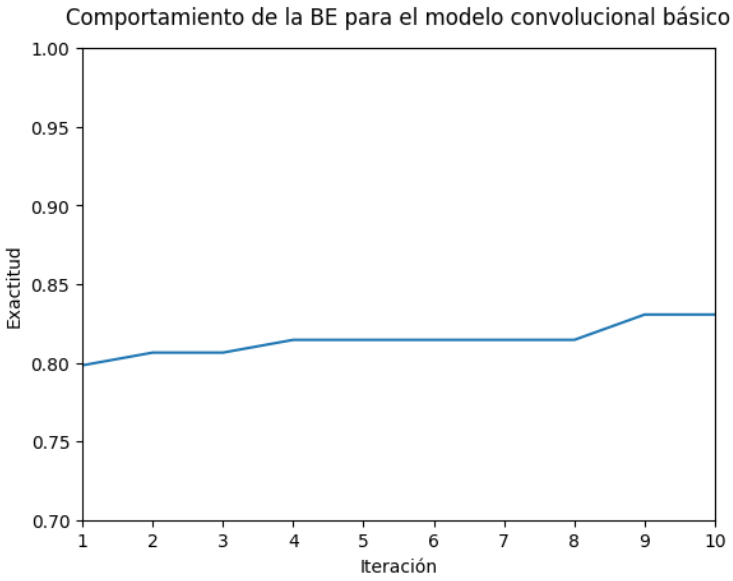


Figura 4.13: Comportamiento de la optimización del modelo convolucional básico con el algoritmo DE durante la BE

iteración y llegando a 81.45 % en la iteración cuatro. Durante las siguientes cuatro iteraciones no se obtiene mejora alguna manteniendo un comportamiento constante. Finalmente, en la novena iteración se alcanza la máxima exactitud obtenida por esta configuración experimental, correspondiente a 83.06 % de exactitud respecto al conjunto de validación, superando de manera significativa los resultados obtenidos durante la BG y la BF.

En el Cuadro 4.19 se muestra el modelo óptimo encontrado por la BE con el enfoque convolucional básico. La sección de extracción de características se compone de una capa *Input*, para imágenes con dimensión de 224×224, seguida de 4 bloques convolucionales. Cada uno de los bloques cuenta con 416, 223, 53 y 512 filtros de tamaño 3×3 respectivamente y mantienen la estructura de dos capas ya mencionada (es decir, una capa *Conv2D()* con una activación *ReLU* y una capa *MaxPool2D*(2×2)). Para la sección de clasificación, los mapas de características de dimensión (14×14) son aplanados mediante una capa *Flatten()*, seguido de dos capas *Dense()* con activación *ReLU* y su correspondiente número de neuronas (441 y 71) y finalizando con una capa de salida que consta de una capa *Dense()* de dos neuronas y activación *softmax*.

El entrenamiento del modelo fue realizado con un total de 102 épocas con un BS de 15 y utilizando el optimizador SGD con una LR de 0.001 y momentum de 0.9. Finalmente, se obtiene una exactitud de 83.06 % respecto al conjunto de validación. La estructura optimizada obtenida es similar a la obtenida durante la BG, ya que cuenta con el mismo número de bloques convolucionales y sólo una capa densa menos. La principal diferencia se encuentra en la cantidad de filtros y neuronas de la red, lo que hace que sea una red más pesada y

#Épocas	102	
BS	15	
Exactitud	83.06 %	
Nombre de Capa	Descripción	Salida
BC_1	Conv(416, 3×3) MaxPool(2×2)	112×112
BC_2	Conv(223, 3×3) MaxPool(2×2)	56×56
BC_3	Conv(53, 3×3) MaxPool(2×2)	28×28
BC_4	Conv(512, 3×3) MaxPool(2×2)	14×14
Flatten		1×100352
Dense_1	Dense(441, ReLU)	1×225
Dense_2	Dense(71, ReLU)	1×112
Out	Dense(2, softmax)	1×2

Cuadro 4.19: Estructura optimizada del enfoque convolucional básico durante la BE

con mayor numero de parámetros. Sin embargo, el incremento de parámetros en la red no resulta muy significativo para el tiempo de cómputo requerido, además de que se obtiene una ganancia notoria en el rendimiento de clasificación.

4.5.2. Optimización del modelo multiescala

El enfoque del modelo multiescala, consiste en la optimización de una estructura de red mediante el uso de bloques multiescala o bloques MFL diseñados para la red MFL. La teoría correspondiente a este tipo de bloque se describe brevemente en la Sección 2.3.1. En este caso, se toma como referencia la arquitectura original de dicha red con el objetivo de optimizar tanto el número de bloques MFL como la cantidad de filtros para cada uno de ellos. La optimización de los hiperparámetros se realiza utilizando los tres tipos de experimentos definidos en este documento (BG, BF y BE). Las especificaciones de los experimentos mencionados y del enfoque multiescala se presentan en la Sección 3.2.2.

Búsqueda gruesa

La ejecución del proceso de optimización se llevó a cabo durante 6 iteraciones, con una población de 15 individuos para cada una. La duración total del experimento fue de 120 horas con 57 minutos, resultando en un tiempo de ejecución significativamente mayor en comparación con los experimentos de BG presentados en apartados anteriores. Este aumento se debe a las estructuras generadas, pues, a pesar de llegar a tener un máximo de tan solo 7 bloques MFL, estos bloques son más complejos que una capa convolucional simple, por lo que las estructuras más pesadas tendrán un tiempo de entrenamiento mayor que cualquiera

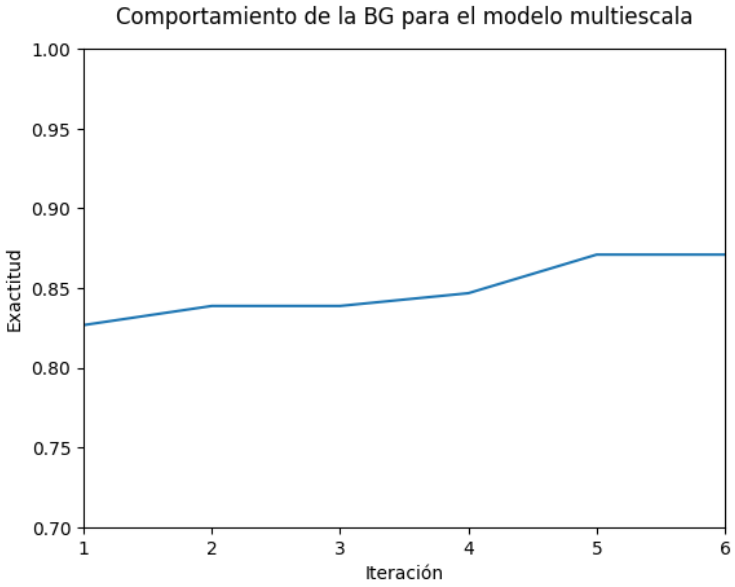


Figura 4.14: Comportamiento de la optimización del enfoque multiescala con el algoritmo DE durante la BG

de las estructuras generadas por el enfoque convolucional básico e incluso que la arquitectura *Resnext50_32x4d*.

En la Figura 4.14 se muestra el comportamiento de la optimización con la BG. Se puede observar que, una estructura conformada por los bloques multiescala de la red MFL es capaz de adaptarse de mejor manera al problema de clasificación de COVID-19 que el enfoque convolucional básico. Los resultados obtenidos muestran que, el proceso de optimización inicia en un valor más alto que los experimentos con el enfoque convolucional básico, alcanzando una exactitud de 82.67 %, mostrando de esta forma un mejor comportamiento desde el inicio. Además, se pueden observar tres puntos de mejora durante la optimización. El primero se presenta en la segunda iteración, donde se alcanza una exactitud de 83.87 %. El segundo corresponde a una ligera mejora de rendimiento, llegando a obtener una exactitud de 84.67 %. Finalmente, el último y más significativo se presenta en la iteración 5, donde se obtiene el máximo rendimiento obtenido con esta configuración experimental, correspondiente a una exactitud de 87.09 % respecto al conjunto de validación.

El modelo óptimo encontrado por la BG con el enfoque multiescala se muestra en el Cuadro 4.20. La estructura de la red optimizada conserva la misma cabecera de red que la versión MFL original, esto es, una capa *Input()* para imágenes con dimensión de 224×224, seguida de un *mini_bolck* y una capa *MaxPool2D*(2×2). El cuerpo de la red se compone de 4 bloques MFL (con 8, 128, 324 y 222 filtros respectivamente), cada uno seguido de una capa *MaxPool2D*(2×2), a excepción del último bloque. Siguiendo el esquema de la red *MFL_Net*, el apartado de clasificación está conformado por una capa *GlobalAveragePooling2D*, una

#Épocas	98	
BS	10	
Exactitud	87.09 %	
Nombre de Capa	Descripción	Salida
Mini_Block	mini_block(16, 3×3)	224×224
Pooling	MaxPool(2×2)	112×112
BMFL_1	MFL_Block(8, 3×3)	112×112
Pooling_1	MaxPool(2×2)	56×56
BMFL_2	MFL_Block(128, 3×3)	56×56
Pooling_2	MaxPool(2×2)	28×28
BMFL_3	MFL_Block(324, 3×3)	28×28
Pooling_3	MaxPool(2×2)	14×14
BMFL_4	MFL_Block(222, 3×3)	14×14
GlobalAveragePooling		1×222
Dropout_1	Dropout(0.5)	1×222
Out	Dense(2, softmax)	1×2

Cuadro 4.20: Estructura optimizada del enfoque multiescala durante la BG

capa de abandono con probabilidad de 50 % y la capa de salida *Dense()* de dos neuronas y activación *softmax*.

En cuanto al entrenamiento, el modelo fue entrenado por 98 épocas con un BS de 10 y utilizando el optimizador SGD con una LR de 0.001 y momentum de 0.9. De esta manera se obtiene una exactitud de 87.09 % respecto al conjunto de validación.

Búsqueda fina

La BF con el enfoque multiescala se realiza alrededor de los resultados obtenidos durante la BG . Sin embargo, al igual que el enfoque convolucional básico, no es posible utilizar el modelo obtenido durante la BG para impulsar el rendimiento de clasificación debido a las diferencias entre las estructuras de red. Para cada uno de los hiperparámetros a optimizar se utilizan los siguientes rangos de búsqueda: [3 : 5] para el numero de bloques MFL; [1 : 58], [78 : 178], [274 : 374], [172 : 272] y [1 : 512] para el número de filtros de cada uno de los respectivos bloques MFL, dado que el modelo obtenido en la BG solo cuenta con 4 bloques MFL, el espacio de búsqueda para los filtros del quinto bloque utiliza los limites originales; [5 : 15] para el BS y [88, 108] para el numero de épocas. La optimización de los individuos generados comienza nuevamente desde cero, pues al modificar la estructura de la red no es posible cargar los pesos obtenidos durante la BG.

El tiempo total de ejecución de la optimización fue de 32 horas con 36 minutos, obteniendo una reducción de tiempo considerable respecto al experimento de BG de este mismo enfoque. Dicha reducción de tiempo se debe a que, al limitar el espacio de búsqueda en torno al modelo obtenido por la BG, se desechan la mayoría de las arquitecturas posibles más pe-

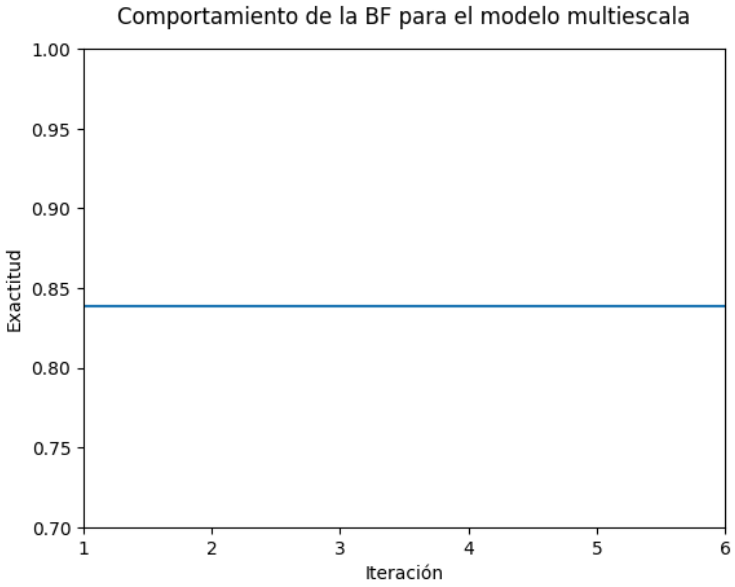


Figura 4.15: Comportamiento de la optimización del enfoque multiescala con el algoritmo DE durante la BF

sadas, que demandan un mayor tiempo de ejecución y que no obtienen un buen rendimiento. Respecto al proceso de optimización, la Figura 4.15 muestra un comportamiento constante. La optimización inicia alcanzando un rendimiento de clasificación de 83.87 % en exactitud, siendo un punto de inicio más alto que el obtenido durante la BG. Sin embargo, ya no se logra mejorar dicho resultado durante en el transcurso del experimento, finalizando con un rendimiento de clasificación significativamente inferior que el obtenido por la BG.

El modelo óptimo encontrado por la BF con el enfoque multiescala se muestra en el Cuadro 4.18. La estructura de la red optimizada presenta múltiples diferencias a lo obtenido en el experimento anterior. Se conserva la misma cabecera de red que consta de una capa *Input()* para imágenes con dimensión de 224×224, un *mini_ bolck* y una capa *MaxPool2D(*2×2*)*. La principal diferencia se encuentra en el cuerpo de la red, teniendo sólo tres bloques MFL con 15, 123 y 341 filtros de tamaño 3×3 cada uno y con su respectiva capa *MaxPool2D(*2×2*)*, a excepción del último bloque. Finalmente, el apartado de clasificación está conformado por una capa *GlobalAveragePooling2D*, una capa de abandono con probabilidad de 50 % y la capa de salida *Dense()* de dos neuronas y activación *softmax*.

El entrenamiento de la estructura obtenida se realizó durante 88 épocas con un BS de 15 y utilizando el optimizador SGD con una LR de 0.001 y momentum de 0.9. En este caso se obtuvo una exactitud de 83.87 % respecto al conjunto de validación. En general se observa que, el entrenamiento de la red se realiza por menos épocas y un BS mayor que el modelo obtenido por la BG, además de contar con un bloque MFL menos. Gracias a estas características, la estructura resultante es más ligera, con menor cantidad de parámetros y

#Épocas	88	
BS	15	
Exactitud	83.87 %	
Nombre de Capa	Descripción	Salida
Mini_Block	mini_block(16, 3×3)	224×224
Pooling	MaxPool(2×2)	112×112
BMFL_1	MFL_Block(15, 3×3)	112×112
Pooling_1	MaxPool(2×2)	56×56
BMFL_2	MFL_Block(123, 3×3)	56×56
Pooling_2	MaxPool(2×2)	28×28
BMFL_3	MFL_Block(341, 3×3)	28×28
GlobalAveragePooling		1×341
Dropout_1	Dropout(0.5)	1×341
Out	Dense(2, softmax)	1×2

Cuadro 4.21: Estructura optimizada del enfoque multiescala durante la BF

un entrenamiento más rápido, pero dichas ventajas no son aceptables como para justificar la significativa reducción de rendimiento.

Búsqueda extensa

La optimización de hiperparámetros de estructura mediante la BE para el enfoque multiescala se llevó a cabo durante 10 iteraciones con una población de 30 individuos para cada una. La duración total del experimento fue de 267 horas con 57 minutos, resultando en un tiempo de ejecución significativamente superior al de cualquiera de los experimentos anteriormente realizados. El incremento en el tiempo de ejecución se debe a la complejidad de los bloques multiescala y la gran cantidad de parámetros de los mismos es consecuencia de trabajar con las cantidades de filtros más altas disponibles en el espacio de búsqueda. Ésto, provoca un incremento considerable en el tiempo de entrenamiento de cada modelo generado, pudiendo llegar a superar al tiempo de entrenamiento de cualquiera de los modelos utilizados en experimentos anteriores.

El comportamiento de la optimización con la BE se muestra en la Figura 4.16. El proceso de optimización inicia obteniendo un rendimiento 83.06 % de exactitud, acercándose desde un comienzo al mejor resultado obtenido durante la BF e iniciando en un punto ligeramente mejor que la BG; esto se logra por el aumento del tamaño de la población. Tan sólo en la segunda iteración, se logra alcanzar una exactitud de 84.67 %, superando al resultado obtenido durante la BF. En la cuarta iteración se presenta otro punto de mejora con una exactitud de 86.29 %, manteniéndose en este punto durante las dos iteraciones siguientes. Finalmente, durante la séptima iteración se logra alcanzar el máximo rendimiento obtenido con esta configuración, correspondiente a un 87.90 % de exactitud respecto al conjunto de validación, superando ligeramente el resultado de la BG.

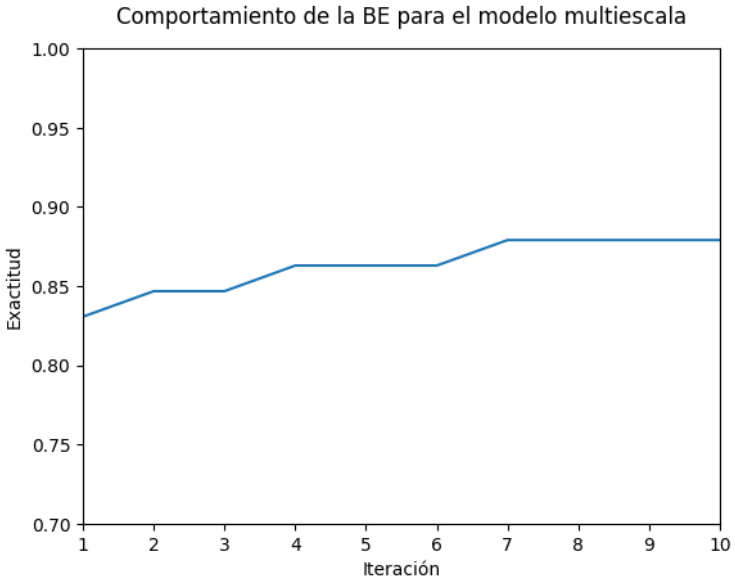


Figura 4.16: Comportamiento de la optimización del enfoque multiescala con el algoritmo DE durante la BE

El modelo óptimo encontrado por la BE con el enfoque multiescala se muestra en el Cuadro 4.22. La estructura de la red optimizada presenta cierta similitud al modelo obtenido en el experimento de BG de este mismo enfoque, específicamente en el número de bloques multiescala que conforman el cuerpo de la red. Sin embargo, se puede observar una gran diferencia en cuanto a la cantidad de filtros a la entrada de cada uno de estos bloques, los cuales corresponden a 149, 35 y 103 y 499 filtros de tamaño 3×3 respectivamente, cada uno con su respectiva capa *MaxPool2D(*2×2*)*, a excepción del último bloque. Al igual que los modelos anteriores de este mismo enfoque, se conserva la misma cabecera de red conformada por de una capa *Input()* para imágenes con dimensión de 224×224, un *mini_ bolck* y una capa *MaxPool2D(*2×2*)*, mientras que el apartado de clasificación está conformado por una capa *GlobalAveragePooling2D*, una capa de abandono con probabilidad de 50 % y la capa de salida *Dense()* de dos neuronas y activación *softmax*. El entrenamiento de la estructura obtenida se realizó durante 130 épocas con un BS de 6 y se utilizó el optimizador SGD con una LR de 0.001 y momentum de 0.9. Finalmente, se obtuvo una exactitud de 87.90 % respecto al conjunto de validación.

La cantidad de filtros utilizados para cada uno de los bloques multiescala contribuye al aumento en la cantidad de parámetros de la red, por lo que la estructura encontrada en este experimento resulta ser la más pesada entre las tres estructuras correspondientes al enfoque multiescala derivando en un tiempo de entrenamiento ligeramente superior, el cual también se ve influenciado por el tamaño tan pequeño de BS. Debido a estos cambios, se logra un aumento en el rendimiento de clasificación y el aumento de parámetros de la red no es tan

#Épocas	130	
BS	6	
Exactitud	87.90 %	
Nombre de Capa	Descripción	Salida
Mini_Block	mini_block(16, 3×3)	224×224
Pooling	MaxPool(2×2)	112×112
BMFL_1	MFL_Block(149, 3×3)	112×112
Pooling_1	MaxPool(2×2)	56×56
BMFL_2	MFL_Block(35, 3×3)	56×56
Pooling_2	MaxPool(2×2)	28×28
BMFL_3	MFL_Block(103, 3×3)	28×28
Pooling_3	MaxPool(2×2)	14×14
BMFL_4	MFL_Block(499, 3×3)	14×14
GlobalAveragePooling		1×499
Dropout_1	Dropout(0.5)	1×499
Out	Dense(2, softmax)	1×2

Cuadro 4.22: Estructura optimizada del enfoque multiescala durante la BE

grande como para que represente un gran impacto en la cantidad de recursos computacionales necesarios para computar la misma. Por lo que se puede considerar que el modelo presentado en el Cuadro 4.22 es el de mejor comportamiento y rendimiento obtenido para el enfoque multiescala.

4.5.3. Optimización del modelo residual

La optimización con el enfoque del modelo residual, consiste en la optimización de una estructura de red mediante el uso de bloques residuales de cuello de botella diseñados para la familia de redes *Resnet*. La teoría correspondiente a este tipo de bloque y las redes residuales se explica brevemente en la Sección 2.3.1. Para generar las estructuras durante la optimización, se toma como referencia la plantilla de construcción de las redes residuales, pero únicamente se utilizan bloques de cuello de botella con el objetivo de optimizar tanto el número de bloques como la cantidad de filtros a la entrada de cada uno de ellos. La optimización de los hiperparámetros se realiza utilizando los tres tipos de experimentos definidos en este documento (BG, BF y BE). Las especificaciones de los experimentos mencionados y del enfoque residual se presentan en la Sección 3.2.2.

Búsqueda gruesa

La ejecución del proceso de optimización con la BG para el enfoque residual se llevó a cabo durante 6 iteraciones, con una población de 15 individuos para cada una. La duración total del experimento fue de 10 horas con 53 minutos, resultando en un tiempo de ejecución

significativamente bajo en comparación con los otros enfoques para la optimización de hiperparámetros de estructura. El bajo tiempo de ejecución se debe a la composición de las estructuras generadas, pues, el enfoque residual se inspira en la arquitectura de las redes residuales pero sin utilizar la división de etapas, las cuales anidan diferente cantidad de bloques de cuello de botella por etapa. Este enfoque únicamente utiliza la concatenación de bloques de cuello de botella hasta un máximo de 7, por lo que incluso las estructuras más pesadas tendrán un tiempo de entrenamiento menor que la red *Resnext50_32x4d* (utilizada en la optimización de hiperparámetros de entrenamiento) e incluso que la mayoría de estructuras generadas por los enfoques anteriores.

En la Figura 4.17 se muestra el comportamiento de la optimización con la BG para el enfoque residual. A primera vista, el comportamiento de la optimización parece tener un comportamiento constante ya que las mejoras en el rendimiento de clasificación se presentan únicamente en las primeras iteraciones. Sin embargo, cuenta con dos puntos de mejora para el modelo encontrado, la BG inicia alcanzando una exactitud de 77.32 % durante la primera iteración, dicho rendimiento mejora durante la iteración 2 hasta un 78.22 % de exactitud y la ultima mejora se obtiene durante la tercera iteración con una exactitud final de 79.83 % respecto al conjunto de validación. Los resultados obtenidos se asemejan mucho a lo observado durante la BG del enfoque convolucional básico, teniendo un punto de inicio muy cercano y la misma exactitud final, pero con la ventaja de un tiempo de optimización mucho menor y una convergencia más rápida. Por lo que, el uso de una estructura conformada por los bloques residuales de cuello de botella parece adaptarse de mejor manera al problema de clasificación de COVID-19 que el enfoque convolucional básico.

En el Cuadro 4.23 se muestra el modelo óptimo encontrado por la BG para el enfoque residual. La estructura de la red optimizada conserva la misma cabecera de red utilizada en las arquitecturas de las redes *Resnet*, la cual está conformada por una capa *Conv2D()* de 64 filtros de tamaño 7×7 y con entrada para imágenes de dimensión 224×224, seguida de una capa *MaxPool2D(*3×3*)* con paso dos. La sección de extracción de características se compone de 5 bloques residuales de cuello de botella (BRes_x), cada uno de ellos con 166, 320, 511, 186 y 131 filtros a su entrada. La sección de clasificación de la red inicia con una trasformación de los mapas de características por medio de una capa *GlobalAveragePooling2D* y finaliza con una capa de salida *Dense()* de dos neuronas y activación *softmax*.

En cuanto al entrenamiento, el modelo fue entrenado por 40 épocas con un BS de 23 y utilizando el optimizador SGD con una LR de 0.001 y momentum de 0.9. De esta manera, se obtuvo una exactitud de 79.83 % respecto al conjunto de validación. A pesar de obtener un rendimiento similar, la estructura del presente modelo resulta ser más pesada que la obtenida durante la BG con el enfoque convolucional básico, esto debido al número de bloques residuales resultantes y la complejidad de los mismos. Sin embargo, se puede observar que el número de épocas necesarios para alcanzar tal rendimiento de clasificación es mucho menor con el enfoque residual, plasmando un comportamiento superior de los bloques residuales sobre los del enfoque convolucional básico. Por otra parte, tanto el enfoque convolucional básico como

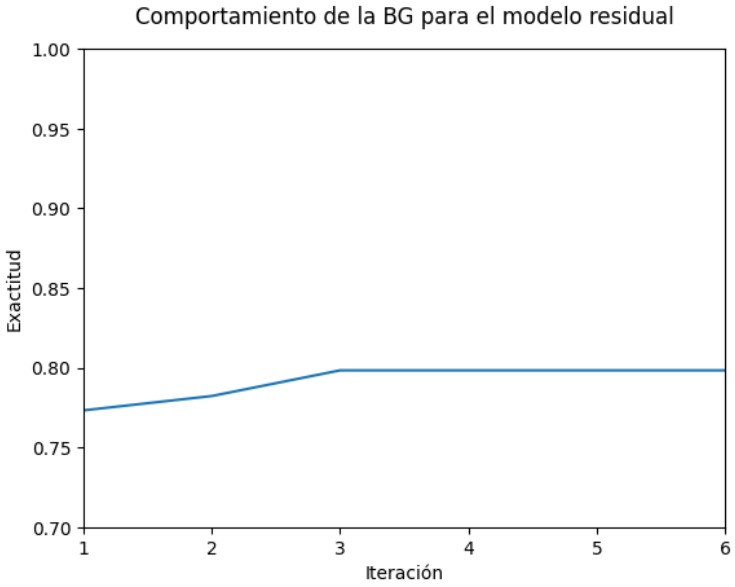


Figura 4.17: Comportamiento de la optimización del enfoque residual con el algoritmo DE durante la BG

el residual muestran no ser capaces de igualar el rendimiento del enfoque multiescala.

Búsqueda fina

Para el experimento de BF con el enfoque residual, se utilizan 6 iteraciones con una población de 15 individuos en cada una. La búsqueda para los valores de los hiperparámetros se realiza alrededor de los encontrados durante la BG. Específicamente, se utilizan los siguientes espacios de búsqueda: [4 : 6] para el número de bloques residuales de cuello de botella; [116 : 216], [270 : 370], [461 : 512], [136 : 236], [81 : 181] y [1 : 512] para el número de filtros de cada uno de los respectivos bloques (dado que el modelo obtenido en la BG solo cuenta con 5 bloques residuales, el espacio de búsqueda para los filtros del sexto bloque utiliza los límites originales); [18 : 28] para el BS; y [30, 50] para el numero de épocas. La optimización de los individuos generados comienza desde cero, pues no es posible cargar los pesos obtenidos durante la BG al modificar la estructura de la red.

El tiempo total de ejecución de la optimización fue de 6 horas con 14 minutos, siendo un tiempo considerablemente menor al experimento de la BG de este mismo enfoque. Dicha reducción de tiempo se obtiene gracias a la limitación del espacio de búsqueda, desechando la mayoría de las arquitecturas pesadas que demandan un mayor tiempo de ejecución consideradas durante la BG. Respecto al proceso de optimización, la Figura 4.18 muestra un comportamiento casi constante, lo cual es un comportamiento muy similar para la BF con los tres diferentes enfoques. La optimización inicia alcanzando un rendimiento de clasificación

#Épocas	40	
BS	23	
Exactitud	79.83 %	
Nombre de Capa	Descripción	Salida
conv_in	Conv(64, 7×7)	112×112
Pooling	MaxPool(3×3)	56×56
BRes_1	$\begin{bmatrix} 1 \times 1, 166 \\ 3 \times 3, 166 \\ 1 \times 1, 332 \end{bmatrix}$	28×28
BRes_2	$\begin{bmatrix} 1 \times 1, 320 \\ 3 \times 3, 320 \\ 1 \times 1, 640 \end{bmatrix}$	14×14
BRes_3	$\begin{bmatrix} 1 \times 1, 511 \\ 3 \times 3, 511 \\ 1 \times 1, 1022 \end{bmatrix}$	7×7
BRes_4	$\begin{bmatrix} 1 \times 1, 186 \\ 3 \times 3, 186 \\ 1 \times 1, 372 \end{bmatrix}$	4×4
BRes_5	$\begin{bmatrix} 1 \times 1, 131 \\ 3 \times 3, 131 \\ 1 \times 1, 262 \end{bmatrix}$	2×2
GlobalAveragePooling		1×524
Out	Dense(2, softmax)	1×2

Cuadro 4.23: Estructura optimizada del enfoque residual durante la BG

de 77.41 % de exactitud, siendo un punto de inicio muy similar al obtenido durante la BG. Sin embargo, solo se presenta un único punto de mejora durante la iteración 2, alcanzando un rendimiento final de 79.83 % en exactitud respecto al conjunto de validación. Es importante destacar que, aún cuando el comportamiento de la BF es más constante que el de la BG (teniendo menos puntos de mejora), el resultado final es muy cercano, obteniendo en ambos experimentos el mismo rendimiento de clasificación final.

El modelo óptimo encontrado por la BF con el enfoque residual se muestra en el Cuadro 4.24. La estructura de la red optimizada es muy similar al obtenido durante el experimento anterior. Se conserva la misma cabecera de red que consta de una capa *Conv2D()* con 64 filtros de tamaño 7×7 y una capa *MaxPool2D(*3×3*)* con paso dos. De igual manera, se tiene el mismo número de bloques residuales de cuello de botella y la principal diferencia se encuentra en el número de filtros de cada bloque, siendo 178, 294, 495, 190 y 123 a la entrada de cada uno. Finalmente, el apartado de clasificación está conformado por una capa *GlobalAveragePooling2D*, resultando en un vector ligeramente más pequeño. Por último, la capa de salida *Dense()* tiene dos neuronas y una activación *softmax*. Como ya se mencionó,

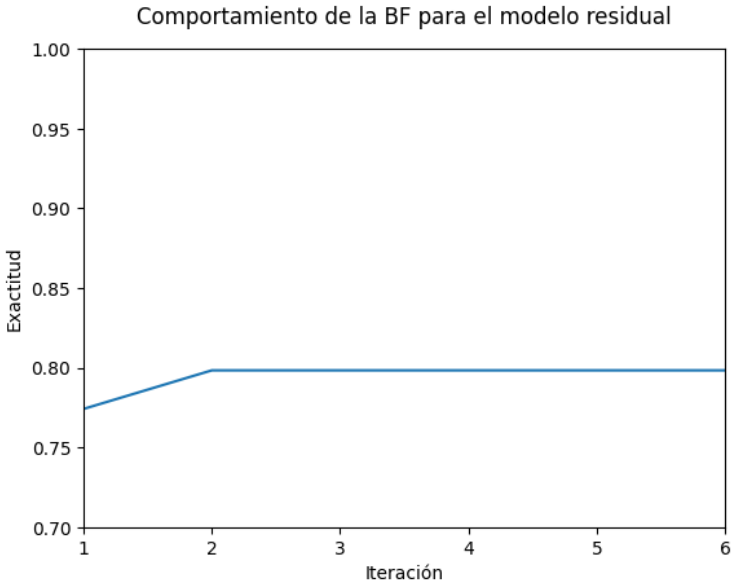


Figura 4.18: Comportamiento de la optimización del enfoque residual con el algoritmo DE durante la BF

a excepción del número de filtros, la estructura de la red obtenida durante la BF es muy similar a la obtenida durante la BG incluso en la cantidad de parámetros, por lo que es lógico esperar un rendimiento de clasificación similar.

En cuanto al entrenamiento del modelo, se realizó durante 48 épocas con un BS de 22 y utilizando el optimizador SGD con una LR de 0.001 y momentum de 0.9.

Búsqueda extensa

La optimización de hiperparámetros de estructura mediante la BE para el enfoque residual se llevó a cabo durante 10 iteraciones con una población de 30 individuos para cada una. La duración total del experimento fue de 62 horas con 12 minutos, resultando en el experimento de BE más rápido para la optimización de hiperparámetros de estructura. El tiempo de ejecución obtenido es muy similar al mostrado durante los experimentos de BE de la optimización de hiperparámetros de entrenamiento, lo cual resulta lógico, ya que se utiliza un tipo de bloque residual muy similar al empleado en la arquitectura *Resnext50_32x4d*, generando estructuras con una cantidad de parámetros parecida pero de mucho menor profundidad. A pesar de la reducción en la profundidad, el número de filtros manejados para la entrada de cada bloque y los tamaños de BS tan pequeños disponibles pueden generar estructuras que se acercan mucho al tiempo de entrenamiento requerido para una arquitectura de red *Resnext50_32x4d*.

El proceso de optimización se muestra en la Figura 4.19, el cual inicia obteniendo un

#Épocas	48	
BS	22	
Exactitud	79.83 %	
Nombre de Capa	Descripción	Salida
conv_in	Conv(64, 7×7)	112×112
Pooling	MaxPool(3×3)	56×56
BRes_1	$\begin{bmatrix} 1 \times 1, 178 \\ 3 \times 3, 178 \\ 1 \times 1, 332 \end{bmatrix}$	28×28
BRes_2	$\begin{bmatrix} 1 \times 1, 294 \\ 3 \times 3, 294 \\ 1 \times 1, 640 \end{bmatrix}$	14×14
BRes_3	$\begin{bmatrix} 1 \times 1, 495 \\ 3 \times 3, 495 \\ 1 \times 1, 1022 \end{bmatrix}$	7×7
BRes_4	$\begin{bmatrix} 1 \times 1, 190 \\ 3 \times 3, 190 \\ 1 \times 1, 372 \end{bmatrix}$	4×4
BRes_5	$\begin{bmatrix} 1 \times 1, 123 \\ 3 \times 3, 123 \\ 1 \times 1, 262 \end{bmatrix}$	2×2
GlobalAveragePooling		1×492
Out	Dense(2, softmax)	1×2

Cuadro 4.24: Estructura optimizada del enfoque residual durante la BF

rendimiento 79.83 % de exactitud, equiparando desde un comienzo a lo obtenido tanto en la BF como la BG de este mismo enfoque. La mejora obtenida desde un punto tan temprano en la optimización se debe principalmente al aumento de la diversidad de información proporcionada por el aumento en el tamaño de la población. Dicho resultado es mejorado durante la segunda iteración de la ejecución, obteniendo una exactitud de 80.64 %. Finalmente, el máximo rendimiento de clasificación se presenta en la cuarta iteración, obteniendo un resultado final de 81.45 % en exactitud respecto al conjunto de validación. Como se muestra en la Figura 4.19, la BE para el enfoque residual presenta un comportamiento más constante en comparación con las BE de los otros enfoques. A pesar de ello, se mantiene congruente al obtener los mejores resultados para su enfoque correspondiente.

En el Cuadro 4.25 se muestra el modelo óptimo encontrado por la BE con el enfoque residual. La estructura de red obtenida se compone por la cabecera de red de las arquitecturas residuales (ResNet), esto es, una capa *Conv2D()* con 64 filtros de 7×7 seguida de una capa *MaxPool2D(*3×3*)*. El cuerpo de la estructura se compone de dos bloques residuales de cuello de botella, cada uno con 480 y 152 filtros a su entrada. Finalmente, el apartado de clasificación

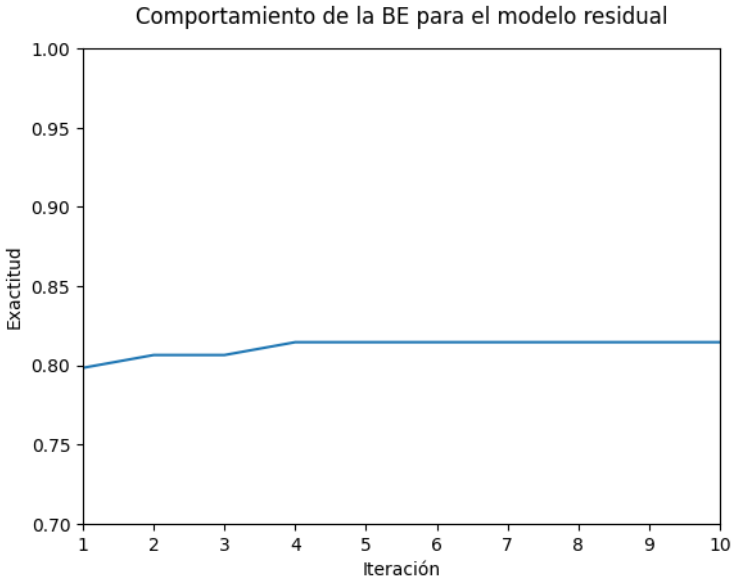


Figura 4.19: Comportamiento de la optimización del enfoque residual con el algoritmo DE durante la BE

está conformado por una capa *GlobalAveragePooling2D* y la capa de salida *Dense()* de dos neuronas y activación *softmax*.

El entrenamiento se realizó durante 113 épocas con un BS de 22 y utilizando el optimizador SGD con una LR de 0.001 y momentum de 0.9. Al ser una estructura con únicamente dos bloques residuales, el modelo obtenido durante este experimento es el más ligero de entre los tres modelos obtenidos para el enfoque residual. Además, éste logra superar el rendimiento de clasificación obtenido tanto en la BG como en la BF, resultando en el modelo que mejor se adapta al problema de caseificación de COVID-19 para dicho enfoque. Sin embargo, el resultado obtenido queda considerablemente alejado del mostrado durante la BE con el enfoque multiescala.

4.5.4. Validación de resultados

Para obtener una evaluación más objetiva de los resultados mostrados, se utilizó el mejor modelo obtenido mediante la optimización de hiperparámetros de estructura para evaluar el rendimiento de clasificación respecto al conjunto de prueba. El modelo seleccionado corresponde al obtenido mediante el experimento de BE con el enfoque multiescala, el cual obtuvo un rendimiento de clasificación de 87.90 % de exactitud respecto al conjunto de validación. Para realizar dicho experimento, se fusionó el conjunto de entrenamiento y validación, utilizados durante el proceso de optimización, para crear el nuevo conjunto de entrenamiento. La estructura de la red optimizada se entrenó durante 130 épocas con un BS de 6, usando el

#Épocas	113	
BS	22	
Exactitud	81.45 %	
Nombre de Capa	Descripción	Salida
conv_in	Conv(64, 7×7)	112×112
Pooling	MaxPool(3×3)	56×56
BRes_1	$\begin{bmatrix} 1 \times 1, 480 \\ 3 \times 3, 480 \\ 1 \times 1, 960 \end{bmatrix}$	28×28
BRes_2	$\begin{bmatrix} 1 \times 1, 152 \\ 3 \times 3, 152 \\ 1 \times 1, 304 \end{bmatrix}$	14×14
GlobalAveragePooling		1×492
Out	Dense(2, softmax)	1×2

Cuadro 4.25: Estructura optimizada del enfoque residual durante la BE

optimizador SGD con una LR de 0.001 y momentum de 0.9 y para la inicialización de pesos se utilizó el archivo del modelo obtenido durante la optimización.

En el Cuadro 4.26 se muestra el rendimiento de clasificación obtenido respecto al conjunto de prueba, en el que se realiza una comparación de rendimientos de clasificación entre el modelo obtenido durante la optimización (modelo multiescala+DE_BE), el modelo de la arquitectura *MFL_Net* original y el modelo *COVID-SDNet*, originario del trabajo donde se presentó el conjunto COVIDGR. En este cuadro se puede observar una ligera disminución de rendimiento de clasificación del modelo multiescala+DE_BE respecto a lo obtenido durante la optimización con el conjunto de validación; dicha disminución es de esperarse, ya que el conjunto de prueba está conformado por datos completamente desconocidos para el modelo. En general, el rendimiento de clasificación obtenido por la arquitectura optimizada logra superar de manera significativa al obtenido por la red MFL original hasta un 10 % en la mayoría de medidas de rendimiento utilizadas. De igual manera, se logra una ligera mejora de rendimiento respecto a las medidas reportadas del modelo *COVID-SDNet*, haciendo énfasis de mejoría en la medida de sensibilidad, la cual indica en este caso, una mejor capacidad del clasificador optimizado para detectar individuos con la enfermedad de COVID-19..

Al igual que en los experimentos presentados de optimización de hiperparámetros de entrenamiento, se decidió realizar una CV k_fold con el objetivo de obtener una mejor estimación del comportamiento del modelo multiescala+DE_BE. Para esto, se utilizó una configuración 5_fold sobre el conjunto de entrenamiento y validación para crear los subconjuntos utilizados para entrenar y evaluar dicho modelo. Los resultados de la CV se muestran en el Cuadro 4.27 y se logra observar la variación del rendimiento de clasificación según la distribución de los datos. En general, se obtiene un media de exactitud de 96.45 %, siendo

Modelo de optimización	Exactitud	Precisión	Sensibilidad	Especificidad	F1-score	IBA
Modelo multiescala + DE_BE	82.58	82.60	82.58	82.37	82.51	69.03
MFL_Net	72.07	72.19	72.25	70.85	72.07	50.74
COVID-SDNet	81.00	84.23	76.80	–	80.07	–

Cuadro 4.26: Rendimiento de clasificación con los hiperparámetros optimizados con la BE respecto al conjunto de prueba

Partición	Exactitud
fold_1	94.40
fold_2	99.19
fold_3	93.54
fold_4	97.58
fold_5	97.58
Media	96.45
Std	2.13

Cuadro 4.27: Rendimiento de clasificación obtenido mediante la CV con configuración 5_fold para el modelo *multiescala+DE_BE*

un resultado considerablemente más elevado a lo obtenido durante la optimización. La razón del incremento tan significativo en este apartado es la inicialización de los pesos del modelo, ya que se realiza utilizando el archivo de pesos obtenido durante el proceso de optimización. Por lo que, durante la CV el modelo ya conoce gran parte de los datos de las particiones aleatorias. Obteniendo de esta manera los resultados de clasificación superiores al 90 % de exactitud en cada una de las particiones, con un máximo de 99.19 % en la segunda partición y un mínimo de 93.54 % en la tercera.

Capítulo 5

Conclusiones y trabajo futuro

En el presente trabajo de tesis se han implementado dos estrategias para optimizar, mediante metaheurísticas, el rendimiento de redes neuronales profundas aplicadas a clasificación de COVID-19, utilizando imágenes de XR. En la primera, se realizó la optimización de hiperparámetros de entrenamiento de la red *Resnext50_32x4d* mediante el uso de tres metaheurísticas (GA, DE y GWO). En la segunda estrategia se realizó la optimización de hiperparámetros de estructura mediante la metaheurística DE y considerando tres enfoques distintos para generar dichas estructuras de CNN (convolucional básico, multiescala y residual). La base de datos empleada fue COVIDGR, esto con el objetivo de obtener una comparación objetiva entre los resultados obtenidos.

Para ambas estrategias, se implementaron tres diferentes configuraciones experimentales con cada metaheurística (BG, BF y BE), variando en cada una, el número de iteraciones, tamaño de la población y el espacio de búsqueda. La experimentación realizada muestra que, los experimentos de BE requieren de un mayor tiempo de ejecución debido al aumento en el tamaño de población y el número de iteraciones. En consecuencia, obtienen una mayor diversidad de información, logrando los mejores resultados en la optimización de los modelos. Por otra parte, los experimentos de BF muestran un impulso inicial en el rendimiento de clasificación al utilizar una inicialización de pesos a partir de la optimización, logrando una convergencia más rápida respecto a la BG. Este impulso también es capaz de mejorar el rendimiento de clasificación del modelo, como se muestra durante la evaluación del modelo optimizado con el conjunto de prueba y en algunas arquitecturas de red fijas que utilizan la técnica de TL.

De esta manera, se logró obtener una mejora en el rendimiento de clasificación de COVID-19 con ambas estrategias. Alcanzando un rendimiento de clasificación medio de 88.73 % y un máximo de 90.32 % para la red *Resnext50_32x4d* con la la optimización de hiperparámetros de entrenamiento. Por otro lado, se obtuvo un rendimiento de clasificación de 82.58 % con el enfoque multiescala mediante la optimización de hiperparámetros de estructura. Lo anterior supera el rendimiento de clasificación del modelo *Resnext50_32x4d* con ajuste manual (tomado de la literatura), de la red *MFL_Net* original e incluso al modelo *COVID-SDNet*. Con

los resultados presentados en este trabajo de tesis se concluye que, la optimización mediante metaheurísticas es una alternativa efectiva para encontrar los hiperparámetros de una red neuronal profunda aplicada a clasificación de COVID-19, ahorrando una cantidad de tiempo y esfuerzo humano considerable y al mismo tiempo mejorando el rendimiento de clasificación.

Cabe señalar que, los recursos de cómputo disponibles fueron una limitante para la especificación y características de los diferentes experimentos de optimización realizados, así como para los hiperparámetros seleccionados y cantidad de los mismos. Estas limitantes se deben principalmente a la memoria y la tarjeta aceleradora disponible, causando largos tiempos de ejecución e incluso la interrupción de algunos experimentos de mayor complejidad.

Como trabajo futuro, se podría realizar una experimentación de optimización más extensa que incluya un enfoque combinado entre más hiperparámetros de entrenamiento y estructura, tomando en cuenta aquellos que no fueron considerados en este trabajo, incluyendo otras metaheurísticas. También, se podría extender la experimentación de la optimización de hiperparámetros de estructura utilizando hardware más potente.

Bibliografía

Abbas, A., Abdelsamea, M. M., and Gaber, M. M. (2021). Classification of COVID-19 in chest X-ray images using DeTraC deep convolutional neural network. *Applied Intelligence*, 51(2):854–864.

Aggarwal, C. C. (2018). *Neural networks and deep learning*. Springer.

Aguilar-Gamboa, F. R. (2020). Desafíos para el manejo y detección de pacientes con COVID-19 en Latinoamérica. *Revista Experiencia en Medicina del Hospital Regional Lambayeque*, 6(1):43–50.

Ahmad, F., Farooq, A., and Ghani, M. U. (2021). Deep ensemble model for classification of novel coronavirus in chest x-ray images. *Computational intelligence and neuroscience*, 2021.

Aszemi, N. M. and Dominic, P. (2019). Hyperparameter optimization in convolutional neural network using genetic algorithms. *International Journal of Advanced Computer Science and Applications*, 10(6):269–278.

Calvo, C., López-Hortelano, M. G., de Carlos Vicente, J. C., Vázquez-Martínez, J. L., and AEP (2020). Recomendaciones sobre el manejo clínico de la infección por el ≪nuevo coronavirus≫ SARS-CoV2. Grupo de trabajo de la Asociación Española de Pediatría (AEP). *Anales de pediatría*, 92(4):241–e1–241–e11.

Chandra, T. B., Verma, K., Singh, B. K., Jain, D., and Netam, S. S. (2021). Coronavirus disease (covid-19) detection in chest x-ray images using majority voting based classifier ensemble. *Expert systems with applications*, 165:113909.

Chowdhury, M. E., Rahman, T., Khandakar, A., Mazhar, R., Kadir, M. A., Mahbub, Z. B., Islam, K. R., Khan, M. S., Iqbal, A., Al Emadi, N., et al. (2020). Can AI help in screening viral and COVID-19 pneumonia?. *Institute of Electrical and Electronics Engineers Access*, 8:132665–132676.

Cleverley, J., Piper, J., and Jones, M. M. (2020). The role of chest radiography in confirming covid-19 pneumonia. *British Medical Journal*, 370:1–9.

Cohen, J. P., Morrison, P., Dao, L., Roth, K., Duong, T. Q., and Ghassemi, M. (2020). Covid-19 image data collection: Prospective predictions are the future. *arXiv preprint arXiv:2006.11988*.

Corman, V. M., Landt, O., Kaiser, M., Molenkamp, R., Meijer, A., Chu, D. K., Bleicker, T., Brünink, S., Schneider, J., Schmidt, M. L., Mulders, D. G., Haagmans, B. L., van-der Veer, B., van-den Brink, S., Wijsman, L., Goderski, G., Romette, J. L., Ellis, J., Zambon, M., Peiris, M., Goossens, H., Reusken, C., Koopmans, M. P., and Drosten, C. (2020). Detection of 2019 novel coronavirus (2019-nCoV) by real-time RT-PCR. *Eurosurveillance*, 25(3):1–8.

Díaz, I. R. et al. (2014). Imágenes diagnósticas: conceptos y generalidades. *Revista de la Facultad de Ciencias Médicas*, pages 35–42.

Ghaderzadeh, M. and Asadi, F. (2021). Deep learning in the detection and diagnosis of covid-19 using radiology modalities: a systematic review. *Journal of healthcare engineering*, 2021.

GOBIERNO DE MÉXICO (2023). Covid-19 México. Recuperado Octubre de 2023, de: https://datos.covid-19.conacyt.mx/.

Goel, T., Murugan, R., Mirjalili, S., and Chakrabartty, D. K. (2021). OptCoNet: an optimized convolutional neural network for an automatic diagnosis of COVID-19. *Applied Intelligence*, 51(3):1351–1366.

Gonzalez, R. C. and Woods, R. E. (2008). *Digital image processing*. Prentice Hall.

Gordo, M. P., Weiland, G. B., García, M. G., and Choperena, G. A. (2021). Aspectos radiológicos de la neumonía covid-19: evolución y complicaciones torácicas. *Radiología*, 63(1):74–88.

Guo, Y. R., Cao, Q. D., Hong, Z. S., Tan, Y. Y., Chen, S. D., Jin, H. J., Sen, K. T., Yun, D. W., and Yan, Y. (2020). The origin, transmission and clinical therapies on coronavirus disease 2019 (COVID-19) outbreak – an update on the status. *Military Medical Research*, 7(1):1–10.

Haykin, S. (2007). *Neural networks: a comprehensive foundation*. Prentice Hall.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Heidari, M., Mirniaharikandehei, S., Khuzani, A. Z., Danala, G., Qiu, Y., and Zheng, B. (2020). Improving the performance of CNN to predict the likelihood of COVID-19 using chest X-ray images with preprocessing algorithms. *International journal of medical informatics*, 144:1–9.

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A. R., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T. N., and Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *Institute of Electrical and Electronics Engineers Signal processing magazine*, 29(6):82–97.

Iba, H. (2018). *Evolutionary approach to machine learning and deep neural networks.* Springer.

INEGI (2021). Características de las defunciones registradas en México durante 2020. Recuperado Noviembre de 2021, de: https://www.inegi.org.mx/app/saladeprensa/noticia.html?id=6867.

INEGI (2023). Estadísticas de defunciones registradas (edr) 2022. Recuperado Diciembre de 2023, de: https://www.inegi.org.mx/app/salaDeprensa/noticia.html?id=8548.

Irmak, E. (2021). Covid-19 disease severity assessment using cnn model. *IET image processing*, 15(8):1814–1824.

Islam, M. M., Karray, F., Alhajj, R., and Zeng, J. (2021). A review on deep learning techniques for the diagnosis of novel coronavirus (covid-19). *IEEE Access*, 9:30551–30572.

Ismael, A. M. and Şengür, A. (2021). Deep learning approaches for covid-19 detection based on chest X-ray images. *Expert Systems with Applications*, 164:114054.

Iwendi, C., Mahboob, K., Khalid, Z., Javed, A. R., and Rizwan, M. (2021). Classification of COVID-19 individuals using adaptive neuro-fuzzy inference system. *Multimedia Systems*, aceptado.

Jangam, E., Dias-Barreto, A. A., and Rao-Annavarapu, C. S. (2021). Automatic detection of COVID-19 from chest CT scan and chest X-Rays images using deep learning, transfer learning and stacking. *Applied Intelligence*, aceptado.

Joshi, A. M. and Nayak, D. R. (2022). Mfl-net: An efficient lightweight multi-scale feature learning cnn for covid-19 diagnosis from ct images. *Institute of Electrical and Electronics Engineers Journal of Biomedical and Health Informatics*, 26(11):5355–5363.

Kaggle (2018). Chest x-ray images (pneumonia). Recuperado Marzo de 2022, de: https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia.

Karar, M. E., Hemdan, E. E.-D., and Shouman, M. A. (2021). Cascaded deep learning classifiers for computer-aided diagnosis of COVID-19 and pneumonia diseases in X-ray scans. *Complex & Intelligent Systems*, 7(1):235–247.

Khan, A., Akram, M. U., and Nazir, S. (2023). Automated grading of chest x-ray images for viral pneumonia with convolutional neural networks ensemble and region of interest localization. *PloS one*, 18(1):e0280352.

Khan, I. U. and Aslam, N. (2020). A deep-learning-based framework for automated diagnosis of COVID-19 using X-ray images. *Information*, 11(9):1–13.

Khuzani, A. Z., Heidari, M., and Shariati, S. A. (2021). Covid-classifier: An automated machine learning model to assist in the diagnosis of COVID-19 infection in chest X-ray images. *Scientific Reports*, 11(1):1–6.

Kiziloluk, S. and Sert, E. (2022). Covid-ccd-net: Covid-19 and colon cancer diagnosis system with optimized cnn hyperparameters using gradient-based optimizer. *Medical & Biological Engineering & Computing*, 60(6):1595–1612.

Kovács, A., Palásti, P., Veréb, D., Bozsik, B., Palkó, A., and Kincses, Z. T. (2021). The sensitivity and specificity of chest CT in the diagnosis of COVID-19. *European Radiology*, 31(5):2819–2824.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105.

Kumar, P. and Hati, A. S. (2021). Deep convolutional neural network based on adaptive gradient optimizer for fault detection in SCIM. *International Society of Automation transactions*, 111:350–359.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the Institute of Electrical and Electronics Engineers*, 86(11):2278–2324.

Li, S., Liu, Y., Sui, X., Chen, C., Tjio, G., Ting, D. S. W., and Goh, R. S. M. (2019). Multi-instance multi-scale cnn for medical image classification. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 531–539.

Lopez-Betancur, D., Bosco Duran, R., Guerrero-Mendez, C., Zambrano-Rodríguez, R., and Saucedo-Anaya, T. (2021). Comparación de arquitecturas de redes neuronales convolucionales para el diagnóstico de covid-19. *Computación y Sistemas*, 25(3):601–615.

López-Ponce de León, J. D., Cárdenas-Marín, P. A., Giraldo-González, G. C., and Escandón, A. H. (2020). Coronavirus - COVID 19: Más allá de la enfermedad pulmonar, qué es y qué sabemos del vínculo con el sistema cardiovascular. *Revista Colombiana de Cardiología*, 27(3):142–152.

Mahdaddi, A., Meshoul, S., and Belguidoum, M. (2021). EA-based hyperparameter optimization of hybrid deep learning models for effective drug-target interactions prediction. *Expert Systems with Applications*, 185:115525.

Manna, S., Wruble, J., Maron, S. Z., Toussie, D., Voutsinas, N., Finkelstein, M., Cedillo, M. A., Diamond, J., Eber, C., Jacobi, A., et al. (2020). Covid-19: a multimodality review of radiologic techniques, clinical utility, and imaging features. *Radiology: Cardiothoracic Imaging*, 2(3):e200210.

Melián, B., Moreno-Pérez, J. A., and Moreno-Vega, J. M. (2003). Metaheurísticas: Una visión global. *Inteligencia Artificial*, 7(19):1–21.

Mirjalili, S., Mirjalili, S. M., and Lewis, A. (2014). Grey wolf optimizer. *Advances in engineering software*, 69:46–61.

Mohimont, L., Chemchem, A., Alin, F., Krajecki, M., and Steffenel, L. A. (2021). Convolutional neural networks and temporal CNNs for covid-19 forecasting in France. *Applied Intelligence*, 51(12):8784–8809.

Muñiz, S. H. and Casanovas, M. M. (2006). Introducción a la tomografía computarizada. *Revista Española de Medicina Nuclear*, 25(3):206–214.

Nagib, A. E., Saeed, M. M., El-Feky, S. F., and Mohamed, A. K. (2022). Hyperparameters optimization of deep convolutional neural network for detecting covid-19 using differential evolution. In *Decision Sciences for COVID-19*, pages 305–325. Springer.

Pal, K. K. and Sudeep, K. S. (2016). Preprocessing for image classification by convolutional neural networks. In *2016 IEEE International Conference on Recent Trends in Electronics, Information and Communication Technology (RTEICT)*, pages 1778–1781.

Panda, A., Naskar, R., Rajbans, S., and Pal, S. (2019). A 3d wide residual network with perceptual loss for brain mri image denoising. In *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pages 1–7.

Perez, D. P., Bustillos, R. S., Botto-Tobar, M., and Mora, C. M. (2021). Análisis de imágenes de rayos X por medio de redes neuronales artificiales. *Ecuadorian Science Journal*, 5(1):55–60.

Perrilliat-García, I., Gámez-Guerrero, M. A., Rocha-Nava, S. L., and Hernández-Oropeza, J. I. (2020). Sistema auxiliar para el diagnóstico de COVID-19 mediante análisis de imágenes de CR torácica basado en Deep Learning. *Memorias del Congreso Nacional de Ingeniería Biomédica*, 7(1):556–563.

Probst, P., Boulesteix, A.-L., and Bischl, B. (2019). Tunability: importance of hyperparameters of machine learning algorithms. *The Journal of Machine Learning Research*, 20(1):1934–1965.

Punn, N. S. and Agarwal, S. (2021). Automated diagnosis of COVID-19 with limited posteroanterior chest X-ray images using fine-tuned deep neural networks. *Applied Intelligence*, 51(5):2689–2702.

Pérez-Abreu, M. R., Gómez-Tejeda, J. J., and Dieguez-Guach, R. A. (2020). Características clínico-epidemiológicas de la COVID-19. *Revista Habanera de Ciencias Médicas*, 19(2):1–15.

Quesada, J. A., López-Pineda, A., Gil-Guillén, V. F., Arriero-Marín, J. M., Gutiérrez, F., and Carratala-Munuera, C. (2021). Período de incubación de la COVID-19: revisión sistemática y metaanálisis. *Revista Clinica Española*, 221(2):109–117.

Restrepo, A. (1998). Procesamiento de imágenes médicas. *Revista Universidad EAFIT*, 34(110):86–92.

Rivas-Asanza, W., Mazón-Olivo, B., and Mejía-Peñafiel, E. (2018). *Generalidades de las redes neuronales artificiales*. Editorial UTMACH.

Rodríguez-Hernández, L. J. and Ochoa-Domínguez, H. d. J. (2021). Métodos para el ajuste de los hiperparámetros de las redes convolucionales. *Memorias de Ciencia y Tecnología*, 1(2):1–2.

Rodríguez-Morales, R. and Sossa-Azuela, J. H. (2012). *Procesamiento y Análisis Digital de Imágenes*. RA-MA S.A. Editorial y Publicaciones.

Sánchez-Caballero, C. C. (2020). *Optimización de hiperparámetros en redes neuronales*. Tesis de ingeniero civil en informática, Pontificia Universisdad Católica de Valparaíso.

Serizawa, T. and Fujita, H. (2020). Optimization of convolutional neural network using the linearly decreasing weight particle swarm optimization. *arXiv preprint arXiv:2001*, 05670:1–10.

Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., and De Freitas, N. (2015). Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the Institute of Electrical and Electronics Engineers*, 104(1):148–175.

Shorten, C. and Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48.

Siddiqi, H. K. and Mehra, M. R. (2020). Covid-19 illness in native and immunosuppressed states: A clinical–therapeutic staging proposal. *The journal of heart and lung transplantation*, 39(5):405–407.

Singh, R. K., Pandey, R., and Babu, R. N. (2021). Covidscreen: explainable deep learning framework for differential diagnosis of covid-19 using chest x-rays. *Neural Computing and Applications*, 33(14):8871–8892.

Song, M., Mallol-Ragolta, A., Parada-Cabaleiro, E., Yang, Z., Liu, S., Ren, Z., Zhao, Z., and Schuller, B. W. (2021). Frustration recognition from speech during game interaction using wide residual networks. *Virtual Reality & Intelligent Hardware*, 3(1):76–86.

Storn, R. and Price, K. (1997). Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the Inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Sánchez-Rivas, M. V. (2020). *La desigualdad social perjudica seriamente la salud: El Coronavirus sí entiende de clases sociales. Reflexiones desconfinadas para la era posCOVID-19*. AnthropiQa.

Tabik, S., Gómez-Ríos, A., Martín-Rodríguez, J. L., Sevillano-García, I., Rey-Area, M., Charte, D., Guirado, E., Suárez, J.-L., Luengo, J., Valero-González, M., et al. (2020). COVIDGR dataset and COVID-SDNet methodology for predicting COVID-19 based on chest x-ray images. *Institute of Electrical and Electronics Engineers journal of biomedical and health informatics*, 24(12):3595–3605.

Tan, Y. S., Lim, K. M., and Lee, C. P. (2021). Wide residual network for vision-based static hand gesture recognition. *IAENG International Journal of Computer Science*, 48(4):76–86.

Viñuela, P. I. and Galván, I. M. (2004). *Redes Neuronales Artificiales. Un Enfoque Práctico*. Prentice Hall.

Wang, L., Lin, Z. Q., and Wong, A. (2020). Covid-net: A tailored deep convolutional neural network design for detection of covid-19 cases from chest x-ray images. *Scientific Reports*, 10(1):1–12.

WHO (2021). Archived: WHO Timeline - COVID-19. Recuperado Octubre de 2021, de: https://www.who.int/news-room/detail/08-04-2020-who-timeline—covid-19.

Woo, S., Park, J., Lee, J.-Y., and Kweon, I. S. (2018). Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19.

worldometers (2021). Covid-19 coronavirus pandemic. Recuperado Octubre de 2021, de: https://www.worldometers.info/coronavirus/?utm_campaign=homeAdvegas1.

Xie, S., Girshick, R., Dollár, P., Tu, Z., and He, K. (2017). Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500.

Xue, Z., You, D., Candemir, S., Jaeger, S., Antani, S., Long, L. R., and Thoma, G. R. (2015). Chest X-ray image view classification. In *2015 IEEE 28th International Symposium on Computer-Based Medical Systems*, pages 66–71.

Yang, H., Gao, L., Tang, N., and Yang, P. (2019). Experimental analysis and evaluation of wide residual networks based agricultural disease identification in smart agriculture system. *EURASIP Journal on Wireless Communications and Networking*, 2019(1):1–10.

Yoo, S. H., Geng, H., Chiu, T. L., Yu, S. K., Cho, D. C., Heo, J., Choi, M. S., Choi, I. H., Van, C. C., Nhung, N. V., Min, B. J., and Lee, H. (2020). Deep learning-based decision-tree classifier for COVID-19 diagnosis from chest X-ray imaging. *Frontiers in medicine*, 7:427.

Zaccato, C. (2020). Coronavirus: los desafíos del mundo del después. América Latina y el impacto de la pandemia del COVID-19. *Pensamiento Propio*, 52:227–235.

Zagoruyko, S. and Komodakis, N. (2016). Wide residual networks. *arXiv preprint arXiv:1605.07146*.

Zhong, Z., Zheng, L., Kang, G., Li, S., and Yang, Y. (2020). Random erasing data augmentation. In *Proceedings of the AAAI conference on artificial intelligence*, pages 13001–13008.

Anexo A

Manual de usuario

En este anexo se indica cómo utilizar la biblioteca desarrollada para realizar la optimización de hiperparámetros de redes neuronales profundas mediante el uso de metaheurísticas, así como el entrenamiento y la evaluación de cada uno de los modelos generados durante la optimización. La estructura general del directorio raíz se muestra en la Figura A.1 y contiene los archivos necesarios para la ejecución del modelo de clasificación de COVID-19, los cuales fueron implementados en el lenguaje de programación Python.

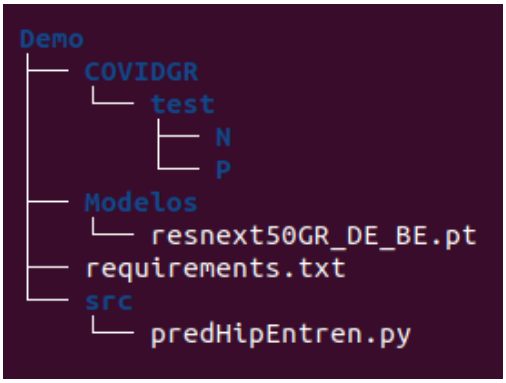


Figura A.1: Estructura del directorio raíz del proyecto

La descripción del contenido del directorio raíz del proyecto es la siguiente:

- La carpeta *COVIDGR* contiene los datos del conjunto de prueba utilizados para realizar la clasificación de COVID-19.
- En la carpeta *Modelos* se incluye un archivo (en formato *pt*) que contiene el mejor modelo de clasificación de COVID-19 obtenido durante la experimentación, el cual corresponde al modelo con la arquitectura *Resnext50_ 32x4d* usando los hiperparámetros obtenidos mediante el experimento BE de la DE.

- La carpeta *src* contiene el script de python utilizado para realizar la clasificación de COVID-19.
- Finalmente, en el archivo *requirements.txt* se encuentra un listado de las dependencias necesarias para la ejecución del script de clasificación.

A.1. Instalación de dependencias

La instalación de las dependencias requeridas es necesaria para asegurar el correcto funcionamiento de la biblioteca desarrollada. Para ésto, se hace uso del Instalador de Paquetes para Python (PIP, por sus siglas en inglés), el cual permite instalar rápidamente bibliotecas Python como pytorch, imblearn, etc. El archivo *requeriments.txt* puede ser utilizado para facilitar el proceso de instalación, el cual contiene un listado de todas las bibliotecas que deben ser instaladas y para iniciar el proceso de instalación basta con ejecutar los comandos mostrados en el Listado B.1.

Listado A.1: Comandos para instalar PIP y las dependencias del proyecto.

- `$ sudo apt install python3 - pip`
- `$ sudo pip3 install -r requirements.txt`

El primer comando instala la herramienta PIP para ser usada con Python3, la cual es necesaria para realizar la instalación de las dependencias contenidas en el archivo de instalación. El segundo comando instala todas estas dependencias para el correcto funcionamiento de la biblioteca del proyecto creada en este trabajo de tesis. Un ejemplo de la descarga e instalación de las dependencias se muestra en la Figura A.2.

A.2. Ejemplo práctico

A continuación, se muestra un ejemplo de la ejecución del script de clasificación de COVID-19, utilizando el mejor modelo obtenido durante la experimentación en este trabajo de tesis. Para iniciar con el proceso de clasificación, únicamente es necesario correr el script de python mediante el siguiente comando:

- `$ python3 ./src/predHipEntren.py`

El comando anterior ejecuta el script de python, el cual se encarga de cargar el modelo y realizar la clasificación de las imágenes del conjunto de prueba, regresando como resultado las medidas de rendimiento, así como la matriz de confusión. En la Figura A.3 se muestra un ejemplo de la salida obtenida en consola.

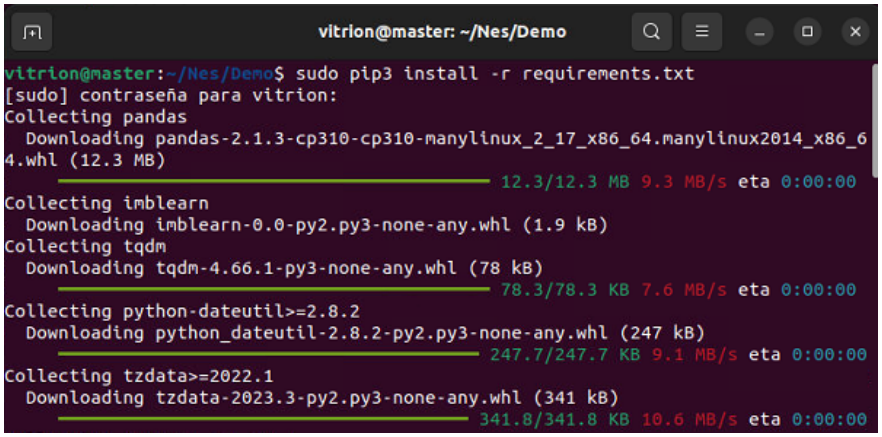


Figura A.2: Ejemplo de la instalación de las dependencias mediante el archivo *requerirments.txt*

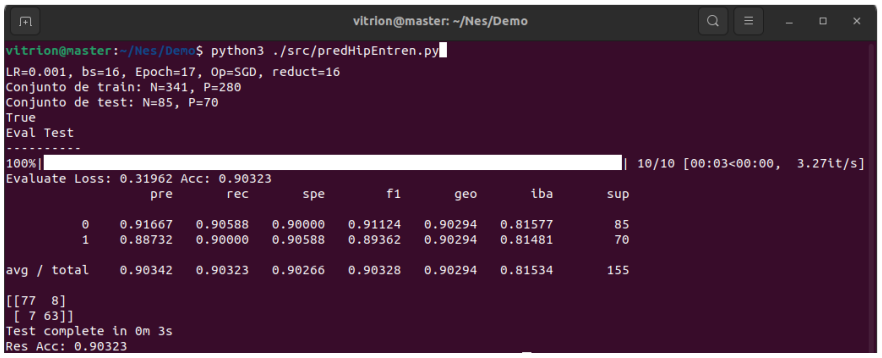


Figura A.3: Ejemplo de la ejecución de la clasificación con el mejor modelo obtenido