UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

PROGRAMA DE MAESTRÍA Y DOCTORADO EN CIENCIAS MATEMÁTICAS Y DE LA ESPECIALIZACIÓN EN ESTADÍSTICA APLICADA

Aprendizaje profundo para la identificación y localización de daños generados por COVID19 en radiografías de tórax.

TESIS
QUE PARA OPTAR POR EL GRADO DE:
MAESTRO EN CIENCIAS

PRESENTA:
JONATHAN JAIR SÁNCHEZ CONTRERAS

DIRECTOR
GIBRAN FUENTES PINEDA IIMAS

CIUDAD DE MÉXICO DICIEMBRE 2024.

Dedicatoria

Soy la convergencia de valiosas aportaciones provenientes de una infinidad de personas que han marcado mi vida con sus enseñanzas, consejos, amor, confianza, entre muchas otras cualidades que han influido en mi visión de la vida, en quién soy y en lo que creo. Siempre estarán presentes en mí, ustedes saben quienes son y lo que significan para mi. Este trabajo lo dedico a cada uno de ustedes.

En palabras de uno de los más grandes científicos de la historia:

"Si he logrado ver más lejos, ha sido porque he subido a hombros de gigantes".
Sir Isaac Newton (1643-1727)

Agradecimientos

Quiero expresar mis más sinceros agradecimientos al Dr. Gibran Fuentes Pineda, quien fungió como mi asesor, por su tiempo, dedicación, consejos, guía y calidad humana. Agradezco la oportunidad de desarrollar el presente trabajo, el cual ha sido crucial para mi crecimiento profesional, personal e intelectual. También extiendo mi agradecimiento a mis mentores y al Instituto de Matemáticas de la UNAM, quienes me brindaron los conocimientos y herramientas necesarios para entender y desarrollar esta tesis, así como para mi vida profesional diaria.

Al Dr. Emilio Marmolejo Olea, del IMATE UNAM, quien fungió como mi tutor durante mis estudios de maestría, gracias por tu apoyo incondicional durante mis estudios y en mi proceso de adaptación como estudiante foráneo en el instituto.

Agradezco al Dr. Christopher David Wood, encargado del Laboratorio Nacional de Microscopía Avanzada del Instituto de Biotecnología UNAM, por su apoyo incondicional como miembro del laboratorio, actividad que me permitió concluir mi tesis.

Al Dr. Adán Oswaldo Guerrero Cárdenas, quien tiene mi total gratitud. Gracias por la confianza inicial y la calidez con la que fui recibido como miembro del Laboratorio Nacional de Microscopía Avanzada.

Agradezco al financiamiento de *The Chan Zuckerberg Initiative* por hacer posible mi estancia de investigación en el Laboratorio Nacional de Microscopía Avanzada, lo cual fue fundamental para mi desarrollo y para la realización de este trabajo, así como para mi participación en otros proyectos.

También agradezco a todos los compañeros del laboratorio, con quienes compartí grandes momentos y quienes tuvieron una influencia profesional y personal enorme.

A CONACYT, agradezco la beca de posgrado que me permitió concluir los créditos del programa de maestría.

Al Programa de Apoyo a Proyectos de Investigación e Innovación Tecnológica (PAPIIT) IV100420 de la UNAM, que impulsó el desarrollo de la presente investigación.

Finalmente, un agradecimiento especial a toda mi familia, en especial a mi mamá, por ser la primera persona en apoyarme, amarme y creer en mí incondicionalmente, y a mis amigos, quienes han sido una fuente constante de cariño, apoyo e inspiración.

Figura 1: La imaginación una frontera difusa entre lo inexistente y lo tangible.

Ilustración realizada por el autor de este trabajo, en la que se plasma una alegoría a la imaginación, la meditación sobre lo que nos hace distintos de una máquina y la indeterminada frontera entre la inteligencia humana y la inteligencia artificial.

Una breve opinión personal sobre el impacto y uso de la inteligencia artificial

La inteligencia artificial ha llegado para ofrecer soluciones y respuestas innovadoras, pero también trae consigo nuevas incógnitas interesantes que nos reconectan como especie con la ardua tarea de descubrirnos a nosotros mismos y lo que nos hace humanos. Emular la inteligencia a través de complejos procedimientos matemáticos nos brinda un avance y una ventana para plantear y resolver cuestionamientos importantes, como la caracterización y comprensión de la conciencia, el libre albedrío, la ética, la moral y la razón. Estas incógnitas, que ocuparon las mentes de los filósofos más brillantes de la historia, parecen estar resurgiendo, invitándonos a meditar sobre aspectos fundamentales de nuestra existencia; aspectos profundos que nos invitan a retomar, fortalecer, valorar, pero sobre todo, a no descuidar la inteligencia natural que reside en cada uno de nosotros.

Es importante recordar que, al ser una herramienta, el beneficio o daño que la inteligencia artificial puede traer a la humanidad, al igual que el fuego en su tiempo, depende de la mano que la empuña y de sus intenciones. Aunque ha surgido el temor de que seamos reemplazados por inteligencias artificiales en múltiples trabajos y actividades, es responsabilidad de la sociedad no permitir que esto suceda. Deberíamos utilizar estas herramientas con cautela, guiados por expertos, para generar una sinergia de habilidades en la resolución de problemas que beneficien a nuestra sociedad.

Así como el descubrimiento de antibióticos o herramientas médicas no desplazó el trabajo del médico, no debemos pretender que la inteligencia artificial tome el papel de un experto. Esto privaría a todas las profesiones de la pasión y creatividad que solo el ser humano puede aportar, lo cual sería una pérdida significativa, en especial para el arte, pues no podemos delegar los procesos creativos que plasman la esencia de la expresión del ser a algo tan poco humano como un computador. Este, al igual que la ciencia, se rige por reglas y metodologías complejas, mientras que el artista no cuenta con ninguna más allá de las que él mismo se impone. De esta manera, la integración de la inteligencia artificial en nuestra sociedad debe verse con una perspectiva similar a la revolución que trajo consigo el internet, transformando nuestras vidas y nuestra forma de aprender, entender, transmitir y recibir información.

El futuro nos ha alcanzado, y con él, los seres humanos hemos desarrollado la inteligencia artificial a nuestra imagen y semejanza. ¿Qué puede significar esto? ¿Acaso es solo una aproximación a una de las características fundamentales e innatas de la humanidad, elegantemente regida por ecuaciones matemáticas que

describen fenómenos probabilísticos y estadísticos, cuidadosamente ejecutadas por una computadora?

La inteligencia artificial ha llegado a la sociedad a través de herramientas tecnológicas creativas que permiten a los usuarios realizar tareas que antes requerían tiempo, esfuerzo o incluso resultaban inimaginables. Posibilita la generación de imágenes realistas, composiciones musicales, videos, edición fotográfica, la redacción y corrección de textos, todo al alcance de un solo clic, de una simple petición bien estructurada en texto o voz.

Esto ha llevado a que las personas adopten diferentes perspectivas sobre estas herramientas, dependiendo del contexto en el que las descubren y utilizan, así como del grado en el que se ven afectados o beneficiados por su uso. Más allá de un uso recreativo, la ciencia se ha visto sumamente beneficiada con el surgimiento de estas herramientas, permitiendo la multidisciplina, la colaboración y el entendimiento de múltiples fenómenos que de otra manera serían inaccesibles.

En áreas críticas para la humanidad, como la biotecnología o la medicina, donde se requiere analizar y procesar cantidades masivas de datos para descubrir correlaciones o dinámicas complejas entre fenómenos, la inteligencia artificial está desempeñando un papel crucial. Ha propiciado el descubrimiento, avance y reinterpretación de lo que se sabía hasta ahora, ofreciendo un cambio de paradigma en múltiples áreas de estudio.

En conclusión, la humanidad está viviendo, a través del surgimiento de la inteligencia artificial, nuevos paradigmas en la percepción de la vida y la realidad. Esto nos plantea una responsabilidad superior como sociedad, ya que nos enfrenta a la posibilidad de desinformación y a noticias falsas sustentadas en evidencias generadas por inteligencias artificiales. Lo anterior ha sucedido a lo largo de la historia con otras tecnologías como el internet, la televisión, la radio, etc.

Más que nunca, el presente nos demanda agudizar nuestra capacidad crítica, nuestra reflexión y la búsqueda de información confiable y verdadera. No debemos sustituir nuestra inteligencia por la inteligencia artificial, pero tampoco negarnos al mundo de respuestas y posibilidades que nos puede brindar. El simple pero profundo acto de pensar ha dotado a la humanidad con su resiliencia de una forma casi artesanal, forjada a través de millones de años de evolución. No podemos dejar esta característica humana esencial a una máquina; sin embargo, el correcto uso de nuevas herramientas siempre ha estado a favor del desarrollo exitoso de nuestra especie.

Resumen

El 31 de diciembre de 2019 se notificó el brote del virus SARS-CoV-2, que posteriormente se convirtió en una pandemia, dando lugar a la crisis de salud global más importante de la actualidad [31]. El SARS-CoV-2, causante de la enfermedad infecciosa conocida como COVID19, se caracteriza por provocar problemas respiratorios y extrarespiratorios de gravedad variable, generando daño y dejando estragos principalmente en los pulmones [32].

El surgimiento de esta enfermedad propició la generación de bases de datos de imágenes médicas con anotaciones específicas realizadas por expertos sobre el daño causado por la infección en los pulmones. Dichas bases de datos han sido utilizadas para entrenar modelos en la detección de daño pulmonar haciendo uso de diversas técnicas de aprendizaje profundo, dando forma a los trabajos que se discuten en el capítulo 4. Esta tesis se enfocó en medir el impacto reflejado en el desempeño al determinar la región espacial donde se ubica la lesión (localización) y la determinación de qué tipo de lesión se trata (identificación), al explorar variaciones en el etiquetado de datos de entrenamiento, elección de hiperparámetros y la aplicación de transferencia de conocimiento para las arquitecturas Y.O.L.O [13] y RetinaNet [66].

La mejora en el desempeño de los modelos brinda a los expertos herramientas tecnológicas más confiables y de acceso público que pueden servir como auxiliar para la detección de daños causados o remanentes de la infección por SARS-CoV-2, permitiendo corroborar diagnósticos, evaluar posibles daños no detectados por el experto o dar seguimiento a pacientes en recuperación. Asimismo, se aporta al acervo tecnológico con herramientas que pueden ser útiles en caso de un resurgimiento del virus o para el estudio de daños causados por nuevas variantes.

Las técnicas exploradas y sus comparativas proporcionan indicadores clave sobre la combinación de hiperparámetros, la calidad y variabilidad de los datos, así como las técnicas de transferencia de conocimiento y aumento de datos, las cuales impactaron positivamente en el entrenamiento de modelos de visión computacional. Además, constituyen un antecedente valioso para futuras investigaciones al considerar la propuesta, entrenamiento y evaluación de nuevos modelos.

Aprendizaje profundo para la identificación y localización de daños generados por COVID19 en radiografías de tórax.

Índice general

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

El avance de la tecnología ha dotado a la humanidad de la capacidad para generar y almacenar grandes volúmenes de información. Al combinarse con disciplinas como las matemáticas, esto permite el análisis masivo de datos, posibilitando el entendimiento de dinámicas complejas y patrones intrínsecos en los mismos. Dicho análisis nos brinda acceso a nuevas respuestas y planteamientos, impulsando avances significativos en diversas áreas, tales como la investigación científica, la optimización de recursos y la economía, que son fundamentales para el desarrollo y bienestar de nuestra especie.

En particular, en el ámbito de la biología y la medicina, los centros de investigación han generado numerosas bases de datos públicas que contienen información con anotaciones detalladas sobre diversas condiciones y características. Estas bases de datos se han convertido en recursos valiosos para la comunidad científica, ya que permiten formular y responder preguntas de manera precisa y eficiente. En el presente trabajo, se utiliza una base de datos pública de imágenes de rayos X de tórax [20], con el objetivo de encontrar, mediante métodos de aprendizaje profundo, estrategias que mejoren el desempeño de los modelos de visión computacional Y.O.L.O [13] y RetinaNet [66] en la tarea de identificar y localizar daños pulmonares ocasionados por la enfermedad infecciosa COVID19.

1.1. Justificación

El COVID19 presenta una tasa de mortalidad más alta que otras enfermedades respiratorias comunes [64]. Sus manifestaciones en radiografías de tórax son muy similares a las de otras neumonías virales y bacterianas, lo que dificulta su diagnóstico [61], resaltando la necesidad de explorar diversas técnicas de diagnóstico e identificación.

Ante esta situación, y utilizando la misma base de datos generada para la competencia pública *SIIM-FISABIO-RSNA COVID-19 Detection*[1] propuesta en la plataforma *Kaggle* por la *Society for Imaging Informatics in Medicine*, cuyo objetivo es la aplicación de modelos de redes neuronales para el mismo propósito planteado en la presente investigación, donde se contó con la participación de 1304 competidores, reflejando los esfuerzos e intereses por mejorar el desempeño de los modelos en la tarea abordada.

[1] La cual se puede consultar en *https://www.kaggle.com/c/siim-covid19-detection/leaderboard*.

Las metodologías comparadas en el artículo titulado *Advancement of deep learning in pneumonia/COVID19 classification and localization: A systematic review with qualitative and quantitative analysis* [54] muestran aplicaciones de aprendizaje profundo con resultados prometedores en diversas tareas relacionadas con el COVID19, distintas a las abordadas aquí, como el diagnóstico de la enfermedad a través de la clasificación de imágenes, la segmentación de áreas dañadas en tomografías computarizadas, entre otras, lo cual alienta al desarrollo de tecnologías enfocadas a nuevas tareas.

Estos antecedentes relevantes motivan a explorar estrategias que mejoren el desempeño de modelos en tareas específicas, como la localización e identificación de daños pulmonares a través de radiografías de tórax, lo cual representa una mejora en términos de costo-beneficio respecto a las técnicas actuales, así como en el acceso a las tecnologías [1].

1.2. Objetivos

Objetivo general

Mejorar el desempeño de los modelos de visión computacional RetinaNet y Y.O.L.O en la tarea de localización e identificación de daño pulmonar ocasionado por COVID19 usando imágenes de rayos X de tórax previamente anotadas por expertos radiólogos.

Objetivos específicos

- Realizar un análisis exploratorio de los conjuntos de datos utilizados en el entrenamiento y la transferencia de conocimiento.
- Preentrenar redes neuronales en tareas secundarias y realizar transferencia de conocimiento para la tarea de localización e identificación de daño pulmonar ocasionado por COVID19.
- Evaluar distintos hiperparámetros, arquitecturas y técnicas de etiquetado para medir su impacto en el desempeño de los modelos.
- Evaluar y comparar el desempeño de los modelos y entrenamientos propuestos.

1.3. Hipótesis

La transferencia de conocimiento, junto con estrategias relacionadas con el aprendizaje profundo, como la elección acertada de hiperparámetros, la exploración respecto al tipo de anotación referente a los datos de entrenamiento y la arquitectura, puede mejorar el desempeño en los modelos de aprendizaje profundo Y.O.L.O [13] y RetinaNet [66] en la localización e identificación de daño en pulmones ocasionados por COVID19 a partir de radiografías de tórax.

Capítulo 2

Conceptos básicos

En este capítulo, se abordan y definen los conceptos e ideas clave utilizados a lo largo del presente trabajo. El objetivo es generar homogeneidad en las definiciones y proporcionar claridad, poniendo a disposición material de consulta dentro del mismo escrito.

2.1. Conceptos biomédicos

Con el propósito de facilitar la comprensión del texto, se proporcionan descripciones generales de los términos y conceptos relacionados con el ámbito médico y de la biología que se utilizan a lo largo del presente trabajo. No se pretende ofrecer una definición formal o rigurosa de los conceptos, más allá de lo necesario para obtener una perspectiva clara y accesible del significado de dichos términos. Esto proporciona sentido al escrito y permite generar una perspectiva adecuada para los lectores sin conocimientos específicos en estas áreas del saber.

2.1.1. Sistema respiratorio

Comencemos por definir el modelo biológico de interés, que consiste principalmente en el aparato respiratorio, entendido como el conjunto de órganos del cuerpo humano involucrados en la respiración. Nos enfocaremos en los componentes donde se presentan los daños causados por el SARS-CoV-2:

- **Pulmón:** Órgano ubicado en el tórax que provee al organismo oxígeno y extrae el dióxido de carbono. Cada pulmón se divide en secciones llamadas lóbulos: tres en el pulmón derecho y dos en el izquierdo.
- **Bronquios:** Conductos principales y más anchos que llevan aire hacia los pulmones. Existe uno para el pulmón izquierdo y otro para el derecho, los cuales permiten el paso del aire hacia los pulmones.
- **Bronquiolos:** Ramificaciones más pequeñas de los conductos aéreos que forman parte de los bronquios y se encargan de dirigir el aire hacia los alvéolos.
- **Alvéolos:** Diminutos sacos de aire ubicados en los extremos de los bronquiolos; en los alvéolos se lleva a cabo el intercambio de oxígeno y dióxido de carbono entre los pulmones y la sangre durante el proceso de respiración.
- **Pleura:** Membrana transparente y muy delgada que recubre los pulmones y reviste el interior de la pared torácica. Permite que los pulmones se muevan suavemente durante la respiración.

Para información detallada sobre los conceptos aquí descritos, véase [36].

2.1.2. Covid19 y su diagnóstico

Los **coronavirus** son un tipo de virus; uno de ellos es el SARS-CoV-2, que produce la enfermedad infecciosa COVID19. Se caracteriza por síntomas que van desde leves, como fiebre y tos, hasta formas más graves que pueden incluir dificultades respiratorias causadas por neumonía. La transmisión principal ocurre a través de gotas respiratorias al toser, estornudar o hablar. La enfermedad fue identificada por primera vez en Wuhan, China, en diciembre de 2019, provocando una pandemia global y dando lugar a importantes esfuerzos para controlar su propagación y desarrollar vacunas para prevenirla. [44]

Uno de los métodos más eficientes y utilizados para el diagnóstico de COVID19 es la prueba de **RT-PCR** [38] o *reverse transcription-polymerase chain reaction* (reacción en cadena de la polimerasa con transcriptasa inversa), la cual es capaz de detectar la presencia de **ADN** o **ARN** [37] del organismo causante de la enfermedad, permitiendo así la detección en las fases más tempranas de la infección.

La prueba se realiza tomando una muestra de material genético que se copia múltiples veces (amplificación). De esta manera, si la muestra contiene organismos infecciosos, se vuelven más fáciles de detectar.

2.1.3. Imágenes médicas

Los médicos recurren al diagnóstico mediante el uso de diversas imágenes del cuerpo humano. En el desarrollo de este proyecto y sus antecedentes, frecuentemente se hace referencia a dos tipos de imágenes médicas:

Radiografías de rayos X: Se obtienen mediante ondas electromagnéticas de rayos X generados por un disco rotatorio de tungsteno bombardeado por electrones, lo que brinda una imagen del interior del cuerpo humano [45], representada en tonos que varían desde el blanco hasta el negro. Esto se debe a que los diferentes componentes del cuerpo tienen una capacidad única de absorción de dichas ondas. Los huesos, ricos en calcio, producen tonos claros, siendo los más visibles debido a su mayor capacidad de absorción. Por otro lado, el aire es el que menos radiación absorbe, generando colores oscuros.

Hay dos tipos de radiografías de rayos X:

- **CR (*Computed Radiography*):** Radiografía computarizada, por su traducción al español, es un formato en el cual se emplea una placa de imágenes de fósforo para crear una imagen digital. Se utiliza un sistema basado en casetes, similar al utilizado en las películas analógicas.
  Es útil cuando se requiere una visión detallada de órganos y estructuras internas, como el cerebro, la columna vertebral, los vasos sanguíneos, entre otros.
- **DX (*Digital Radiography*):** Radiografía digital, por su traducción al español, representa la forma más avanzada en el campo de la radiografía. Emplea

un detector de rayos X digital para adquirir imágenes automáticamente y transferirlas a una computadora para su visualización.
Es eficaz para exámenes rutinarios y de seguimiento destinados a visualizar estructuras menos complejas, como el tórax, las extremidades o los dientes, donde se requiere una imagen rápida y eficiente.

Tomografía computarizada: Se utiliza un equipo médico especial de rayos X para generar una imagen transversal del cuerpo. Su proceso consiste en combinar una serie de radiografías tomadas desde distintos ángulos alrededor del cuerpo, y mediante un procesamiento computacional se construyen imágenes transversales que constituyen una proyección 3D, proporcionando más información y claridad que una radiografía clásica [46].

Debido a la necesidad de almacenar y compartir información digital, ya sea con otros especialistas o con los pacientes, los especialistas se encargaron de establecer **estándares internacionales** para imágenes y texto digital que los centros deben cumplir.

A continuación, se hace mención de uno de los estándares más utilizado.

DICOM: *Digital Imaging and Communications in Medicine* [48], o, por su traducción al español, Imágenes y Comunicaciones Digitales en Medicina, es un estándar utilizado en el ámbito de la medicina para la gestión, almacenamiento, impresión y transmisión de información médica relacionada con imágenes. Las imágenes se producen en la extensión .dcm, un formato independiente del equipo médico utilizado. Este estándar consiste en una imagen de alta resolución diseñada para asegurar la interoperabilidad de las imágenes médicas entre diferentes sistemas y dispositivos médicos, además de un conjunto de metadatos que permite la inclusión de información clínica asociada a la imagen, como datos sobre el paciente, el procedimiento y el dispositivo de adquisición de imágenes, entre otros [47].

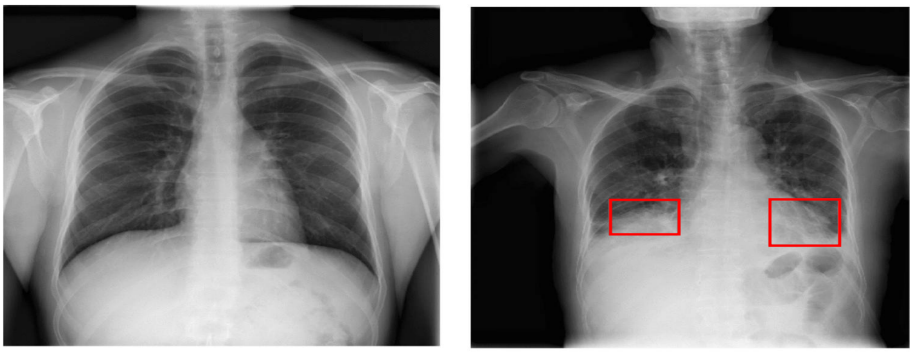
2.1.4. Opacidades en radiografías

Se describe como el fenómeno físico y fisiológico que hace referencia a la capacidad relativa de la materia para obstaculizar la transmisión de energía radiante. Esto se traduce en áreas grises en las imágenes, que son indicadores de la presencia de zonas altamente densas por causas diversas, como se puede observar en la figura 2.1. Las áreas más densas son más opacas, variando en tonalidades grises, mientras que las zonas menos densas son más transparentes y aparecen en tonalidades oscuras. Estas opacidades son de causa variable y cuentan con distintas clasificaciones.

Las opacidades causadas por COVID19, presentes en radiografías de rayos X de pulmón, son los objetos de interés de la presente investigación, donde se busca la localización e identificación automática de estos elementos.

2.2. Imágenes digitales

Dado que las imágenes digitales son uno de los temas centrales del presente trabajo, es fundamental comprender cómo son representadas por las computadoras.



(a) (b)

Figura 2.1: Imagen médica con presencia de opacidades

Imagen tomada de [21]. a) Muestra una imagen de una persona sana, sin presencia de opacidades. b) Imagen que destaca con rectángulos rojos la presencia de opacidades en los pulmones.

2.2.1. Estructura de una imagen digital

Para comprender cómo las computadoras representan las imágenes, es necesario comenzar por entender la estructura básica que permite transformar nuestro concepto basado en la interpretación visual de una imagen al almacenamiento y representación de la información contenida en la misma.

Las computadoras representan las imágenes en blanco y negro utilizando matrices bidimensionales de números. Cada posición en la matriz, denominada *pixel*, almacena un valor numérico que indica la intensidad de gris en esa región específica de la imagen.

El rango de valores utilizados para representar la intensidad depende del formato de la imagen. Por ejemplo, en imágenes de 8 bits, los valores oscilan entre 0 y 255, donde:

- 0 representa el negro absoluto.
- 255 representa el blanco absoluto.
- Los valores intermedios representan diferentes tonalidades de gris.

De esta manera, las imágenes en blanco y negro pueden ser manipuladas mediante operaciones matemáticas sobre sus matrices asociadas, lo que permite aplicar transformaciones como ajustes de contraste, filtros o segmentación.
Esta representación es especialmente útil en procesamiento de imágenes, ya que permite manipular las imágenes utilizando operaciones matriciales y algoritmos numéricos.

Geométricamente, se puede observar en la figura 2.2 un esquema para imágenes de un solo canal, es decir, donde la matriz es un arreglo bidimensional de píxeles, y la imagen representada está en escala de grises.
La **Interpretación fotométrica**: es un concepto referente a la representación de la información relacionada con la intensidad de la luz en una imagen. Se definen dos representaciones básicas.

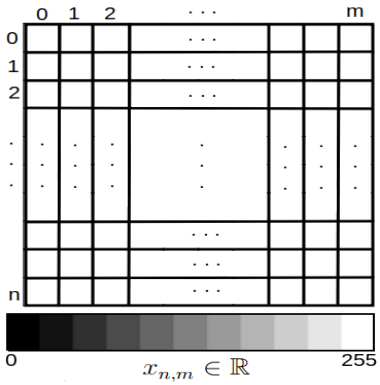


Figura 2.2: Estructura geométrica de una imagen dada como matriz.

Representación geométrica de una imagen con un solo canal de color, donde cada píxel toma valores en el rango $[0, 255]$. Donde la variación del color abarca desde el blanco, pasando por escalas de grises, hasta llegar al negro.

MONOCROMO1: Es una asignación de la escala de intensidades. En este caso, indica que las intensidades van desde las tonalidades más brillantes hasta las más oscuras, describiéndose mediante valores de píxeles ascendentes. Es decir, a medida que el valor de la intensidad del píxel aumenta, el color que representa es más oscuro, como se ilustra en la figura 2.3.

Figura 2.3: MONOCROMO1.

Representación gráfica de MONOCROMO1 donde se muestra la asignación correspondiente para la escala de grises.

MONOCROMO2: Es la versión opuesta de MONOCROMO1; indica que las intensidades van desde las tonalidades más oscuras hasta las más brillantes. A medida que el valor de la intensidad del píxel aumenta, el color que representa es más claro, como se ilustra en la figura 2.4.

Figura 2.4: MONOCROMO2.

Representación gráfica de MONOCROMO2 donde se muestra la asignación correspondiente para la escala de grises.

Para obtener información detallada sobre imágenes digitales, véase [49].

2.3. Inteligencia artificial

La inteligencia artificial (IA) es un campo de la informática centrado en el desarrollo de sistemas capaces de realizar tareas que normalmente requieren inteligencia humana. Estas tareas incluyen el aprendizaje, el razonamiento, la percepción visual, el reconocimiento del lenguaje natural y la toma de decisiones. El objetivo

principal de la IA es crear máquinas que puedan imitar la inteligencia humana para resolver problemas complejos.

Dentro de la inteligencia artificial, el aprendizaje automático o *Machine Learning* (ML) en inglés es una rama esencial. Ofrece técnicas que permiten a las máquinas aprender patrones a partir de datos y mejorar su rendimiento sin intervención humana directa. Para evitar programar explícitamente todas las reglas, los algoritmos de aprendizaje automático permiten que el sistema aprenda de los datos y ajuste su comportamiento en consecuencia, partiendo de un código de programación estático.

En resumen, la inteligencia artificial busca crear sistemas inteligentes que imiten la inteligencia humana. El aprendizaje automático, por otro lado, es una técnica dentro de la inteligencia artificial que permite a las máquinas aprender y mejorar a través de la experiencia. En este contexto, se entiende la inteligencia como la habilidad de aprender a partir de la experiencia y su aplicación en la adaptación del comportamiento. Sin embargo, no se pretende profundizar en los aspectos filosóficos, psicológicos, sociológicos y culturales que implican entender y definir lo que es la inteligencia.

En esta sección, se abordan los conceptos e ideas que facilitan la comprensión y aplicación de la totalidad de metodologías, técnicas y estrategias empleadas a lo largo de la presente investigación.

2.3.1. Conceptos clave

Uno de los conceptos fundamentales para la aplicación de modelos de IA es el **Conjunto de datos**, que es una colección organizada de información utilizada para extraer características de interés. Debe ser altamente representativa de las relaciones que se desean aprender y modelar.

El conjunto de datos desempeña un papel crucial en la implementación de modelos de inteligencia artificial, por lo que debe ser cuidadosamente seleccionado y anotado. Este proceso representa un porcentaje significativo en cuanto al buen desempeño del modelo. Por ende, se debe dedicar una gran parte del tiempo a realizar análisis estadísticos exploratorios para entender las distribuciones y características que están presentes en los datos.

Formalmente, un conjunto de datos anotados se puede entender como una colección de n datos $\mathbb{X} = \{x_1, \ldots, x_n\}$ y su respectivo conjunto de etiquetas o anotaciones $\mathbb{Y} = \{y_1, \ldots, y_n\}$ que describen una relación $\mathscr{D}$, es decir, $\mathscr{D} \subseteq \mathbb{X} \times \mathbb{Y}$, descrita por

$$\mathscr{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$$

Esta relación abstrae la función objetivo de interés y se divide en los siguientes conjuntos

Conjunto de entrenamiento: Porción específicamente destinada para entrenar un modelo. El conjunto de entrenamiento funciona como ejemplos que permiten al modelo ajustar parámetros.

Conjunto de validación: Porción que permite la evaluación del modelo durante el proceso de entrenamiento. Este subconjunto se utiliza después de cada reajuste de parámetros para evaluar la convergencia en el desempeño del modelo y cuantificar la calidad de los parámetros establecidos hasta ese momento.

Conjunto de prueba: Porción utilizada para cuantificar el desempeño final del modelo y su capacidad para generalizar las relaciones aprendidas durante el entrenamiento. Por ende, es importante que ningún elemento de este subconjunto sea utilizado para entrenar o validar el modelo.

Formalmente, el conjunto de datos está dado por la relación $\mathscr{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, donde se define $\mathscr{E}, \mathscr{V}, \mathscr{P} \subset \mathscr{D}$ como la familia de subconjuntos $P = \{\mathscr{E}, \mathscr{V}, \mathscr{P}\}$, que cumple con las siguientes condiciones:

- $\varnothing \notin P$: el conjunto vacío no pertenece a la familia de conjuntos.
- $\bigcup_{A \in P} A = \mathscr{D}$: la unión de la familia de subconjuntos genera el conjunto de datos.
- $\forall A, B \in P$ tal que $A \neq B \implies A \cap B = \varnothing$: los subconjuntos son mutuamente excluyentes.

Es decir, el conjunto de entrenamiento $\mathscr{E}$, el de validación $\mathscr{V}$ y el de prueba $\mathscr{P}$ forman una partición del conjunto de datos. El tamaño de cada uno de estos subconjuntos se define acorde a la cantidad de datos disponibles y la complejidad de la relación que representan.

Por otro lado, un **algoritmo** es un conjunto de instrucciones o reglas lógicas ordenadas, legibles y finitas, diseñadas para realizar una tarea específica. Pueden traducirse a un lenguaje de programación que, al ser compilado, permite a la computadora procesar y ejecutar estas instrucciones.

En el contexto de la inteligencia artificial, los algoritmos definen las instrucciones que permiten a la computadora procesar datos e implementar rutinas de entrenamiento, entre otros procesos que conducen al desarrollo de modelos entrenados en tareas específicas.

El **Entrenamiento** de un modelo es el nombre que se utiliza para especificar el algoritmo que dicta las instrucciones necesarias para que el modelo aprenda las características de interés del conjunto de datos. Esto incluye desde el procesamiento de datos, su estructuración para alimentar el modelo, hasta las técnicas de optimización y evaluación utilizadas para actualizar los parámetros necesarios. Este entrenamiento tiene la finalidad de proporcionar y cuantificar el aprendizaje al modelo.

El **modelo** es el resultado buscado al aplicar un algoritmo de aprendizaje automático. Este abstrae las relaciones representadas en el conjunto al que se aplica, con la finalidad de predecir o generalizar estas relaciones a nuevos datos no etiquetados. Este se compone de la arquitectura utilizada y los parámetros asociados a la misma.

Formalmente, un modelo M con parámetros θ se entiende como una aproximación funcional dada por

$$\mathscr{D} \approx M(x|\theta) : \mathbb{X} \to \mathbb{Y}$$

Respecto a los modelos el **ruido**, también denominado **error irreducible**, es el error siempre presente debido a la naturaleza de la aproximación. Este error se debe, en parte, a las limitaciones computacionales, como redondeos y representaciones numéricas, así como a las discrepancias asociadas a los datos, como errores en los mecanismos de medición, recolección, anotación, entre otros.

Dada la inevitable presencia de este error, siempre debe contemplarse, siendo fundamental definir mecanismos para su minimización.

Formalmente, el ruido está dado por $\hat{y}_i + \varepsilon_i = M(x_i|\theta) + \varepsilon_i$, donde ε_i pertenece a una distribución de probabilidad que depende de la naturaleza de los datos.

2.3.2. Tipos de ajustes

El **ajuste** se refiere a la capacidad de un modelo para adaptarse de manera efectiva a los datos, de modo que pueda realizar predicciones o tomar decisiones precisas en diversas situaciones.

Un **subajuste** ocurre cuando un modelo no logra capturar adecuadamente las complejidades presentes en los datos. Esto suele suceder cuando el modelo es demasiado simple para representar la relación subyacente entre los datos y las etiquetas correspondientes.

Respecto a las curvas de pérdida del entrenamiento y validación son altas y se mantienen cercanas como se ilustra en la figura 2.5, indicando que el modelo no tiene suficiente capacidad para aprender patrones relevantes de los datos.

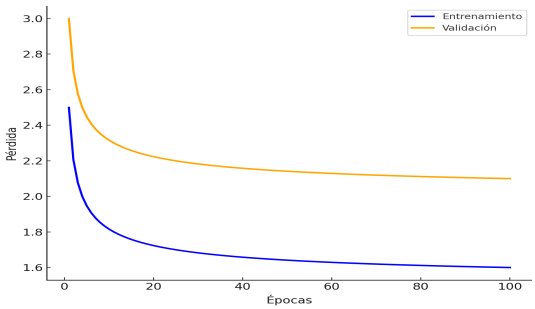


Figura 2.5: Subajuste.

Representación gráfica del comportamiento de las gráficas de entrenamiento en un subajuste.

Por otro lado, un **sobreajuste** se presenta cuando un modelo se ajusta demasiado a los datos, capturando incluso el ruido y las fluctuaciones aleatorias en estos. Esto provoca que el modelo no generalice el aprendizaje a datos nuevos.

Es decir, la gráfica de pérdida de entrenamiento disminuye significativamente, pero la pérdida de validación aumenta después de cierto punto como se observa en la figura 2.6, mostrando que el modelo está memorizando los datos de entrenamiento en lugar de generalizar.

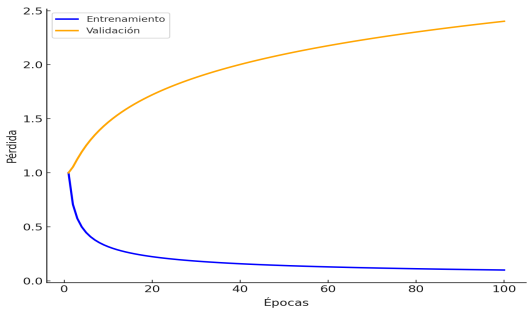


Figura 2.6: Sobreajuste.

Representación gráfica del comportamiento de las gráficas de entrenamiento en un sobreajuste.

En ambos casos, la capacidad de generalización de resultados del modelo se ve afectada. En el subajuste, el modelo no es capaz de hacer predicciones viables, incluso en el conjunto de datos con el que fue entrenado. En el sobreajuste, el modelo puede tener buenos resultados solo en el conjunto de datos utilizado para el entrenamiento.

Para obtener información detallada sobre lo aquí expuesto, consulte [40].

2.3.3. Clasificación de algoritmos

Los algoritmos de aprendizaje automático pueden clasificarse según diversas características, siendo las más comunes el tipo de aprendizaje y el tipo de tarea que desempeñan. En esta sección, se hace referencia al tipo de aprendizaje utilizado para este trabajo.

Aprendizaje supervisado

En este enfoque, se requiere un conjunto de datos previamente etiquetado. Para los fines de esta investigación, se hace énfasis en cuatro tareas que, bajo este enfoque de aprendizaje, son importantes: la clasificación, la regresión, la segmentación y la detección de objetos.

Formalmente, para este tipo de aprendizaje requerimos un conjunto de datos anotado

$$\mathscr{D} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$$

La **clasificación** consiste en algoritmos que asignan categorías a los elementos de un conjunto de datos, con el propósito de hacer predicciones discretas.

Formalmente, la clasificación consiste en algoritmos que permiten aprender la aproximación funcional

$$\mathscr{D} \approx M(x|\theta) : \mathbb{X} \to \mathbb{Y} \text{ donde } x_i \in \mathbb{X} \text{ y } y_i \in \mathbb{Y} \subseteq \mathbb{N}.$$

La **regresión** consiste en algoritmos que permiten aproximar una función entre una variable dependiente y una o más variables independientes, con el propósito de hacer predicciones y estimaciones numéricas continuas.

Formalmente, la regresión consiste en algoritmos que permiten aprender la aproximación funcional

$$\mathscr{D} \approx M(x|\theta) : \mathbb{X} \to \mathbb{Y} \text{ donde } x_i \in \mathbb{X} \text{ y } y_i \in \mathbb{Y} \subseteq \mathbb{R}.$$

La **segmentación** se refiere a algoritmos relacionados con la visión por computadora que se encargan de la tarea de dividir una imagen en subregiones de interés o **regions of interest** (**ROI**), por su traducción al inglés, basadas en ciertos criterios, como la pertenencia a una clase específica. El objetivo de estos algoritmos es asignar una etiqueta a cada píxel de la imagen, indicando a qué clase o categoría pertenece ese píxel, generando máscaras que representan las ROI. La segmentación de regiones de interés es fundamental en aplicaciones que requieren comprender la estructura y contenido visual de las imágenes.

- **Segmentación Semántica:** El objetivo es asignar una etiqueta a cada píxel de la imagen que represente la clase a la que pertenece.
- **Segmentación de Instancias:** Además de asignar una etiqueta a cada píxel, se busca identificar y distinguir cada instancia individual de un objeto dentro de una misma clase.

Finalmente, la **detección de objetos** se refiere a algoritmos relacionados con la visión por computadora que implican localizar e identificar objetos específicos dentro de una imagen o un video, proporcionando información sobre la ubicación precisa de cada objeto mediante una caja delimitadora alrededor de cada objeto presente en la imagen.

Para obtener información detallada sobre la clasificación de modelos aquí expuesta, consulte [41].

2.3.4. Evaluación de modelos

Para determinar la calidad de las predicciones de un modelo de aprendizaje, se requieren métricas que cuantifiquen su desempeño en el aprendizaje y en la generalización de resultados. En esta sección, se mencionan algunas de las métricas más utilizadas.

Una **matriz de confusión** es una herramienta intuitiva que permite determinar distintas métricas respecto a las predicciones de un modelo, tomando en cuenta las etiquetas verdaderas. Se implementa cuando el modelo predice distintas clases, representándose mediante una matriz de $n \times n$, donde n es el número de clases posibles. Esta matriz contiene los conteos de las predicciones posibles, como se ilustra en la figura 2.7.

Verdadero positivo (VP) y Verdadero negativo (VN)

Denotados como **TP** y **TN** por sus respectivas abreviaturas en inglés, son los casos donde la clase predicha coincide con la clase real.
Falso positivo (FP) y Falso negativo (FN)

Denotados como **Error tipo 1** y **Error tipo 2**, respectivamente, son casos donde la clase fue predicha incorrectamente.

Formalmente, una matriz de confusión es un elemento $C \in \mathbb{M}_n(\mathbb{N})$, donde $C_{i,j}$ es el número de instancias de la clase $1 \leqslant i \leqslant n$ que fueron clasificadas como clase $1 \leqslant j \leqslant n$.

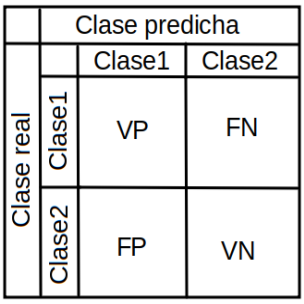


Figura 2.7: Matriz de confusión para $n = 2$ clases.

Representación gráfica de una matriz de confusión para evaluar un modelo de clasificación binaria.

A partir de la matriz de confusión, se definen distintas métricas $\mu : \mathbb{M}_n(\mathbb{N}) \to [0, 1]$ que sirven como indicadores de la calidad de las predicciones del modelo, donde el valor 0 significa un pésimo desempeño y el 1 un excelente desempeño respecto a la métrica definida. Estas están dadas mediante las siguientes formulaciones.

La **exactitud** o ***accuracy***, por su nombre en inglés, representa la proporción de instancias correctamente clasificadas respecto al total de instancias en el conjunto de datos. En otras palabras, mide la fracción de predicciones correctas realizadas por el modelo. Sin embargo, la exactitud puede ser engañosa en situaciones donde las clases están desequilibradas, es decir, cuando alguna de las clases es exageradamente grande o pequeña en comparación con las otras. En esos casos, la exactitud no proporciona una medida representativa del rendimiento del modelo.

$$\text{exactitud} = \frac{VP + VN}{VP + FP + FN + VN}$$

La **precisión** mide la proporción de instancias correctamente clasificadas como positivas entre todas las instancias clasificadas como positivas por el modelo. En otras palabras, la precisión se centra en la calidad de las predicciones positivas del modelo. Proporciona información sobre la proporción de instancias positivas predichas que realmente son positivas. Es útil cuando la importancia de los falsos positivos es alta.

$$\text{precisión} = \frac{VP}{VP + FP}$$

La **sensibilidad** o ***recall***, por su nombre en inglés, mide la proporción de instancias positivas correctamente identificadas por el modelo en relación con el total de instancias que son realmente positivas. En otras palabras, la sensibilidad proporciona una medida de la capacidad del modelo para capturar todas las instancias positivas presentes en el conjunto de datos. Es especialmente útil en situaciones donde la importancia de los falsos negativos es crítica.

$$\text{sensibilidad} = \frac{VP}{VP + FN}$$

La **especificidad** mide la proporción de instancias correctamente clasificadas como negativas entre todas las instancias que son realmente negativas. Se centra en evaluar la capacidad del modelo para identificar correctamente las instancias negativas. Esta métrica es especialmente relevante en situaciones donde la importancia

de los falsos positivos es alta y se busca minimizar la probabilidad de clasificar incorrectamente instancias negativas como positivas.

$$\text{especificidad} = \frac{VN}{VN + FP}$$

Finalmente, el **puntaje F1** es una métrica que combina la precisión y la sensibilidad en un solo valor, proporcionando así una medida más equilibrada. Es particularmente útil cuando hay un desequilibrio entre las clases en el conjunto de datos. Se define como la media armónica de la precisión y la sensibilidad, penalizando más fuertemente los casos en los que una de las dos métricas es baja.

$$F1 = \frac{2\text{Precisión} * \text{Sensibilidad}}{\text{Precisión} + \text{Sensibilidad}}$$

Para obtener información detallada sobre las métricas aquí expuestas, consulte [14].

Por otro lado, el **índice Jaccard** o **IoU** (*Intersection over Union*) es una métrica que mide el grado de similitud entre dos conjuntos, sin importar el tipo de elementos. Se utiliza para evaluar la precisión de las predicciones en problemas de detección de objetos, como se ilustra en la figura 2.8.

Dada una predicción del modelo $M(x_1|\theta) = \hat{y}_1$ y la respectiva etiqueta y_1 para el dato x_1, donde $M(x|\theta)$ es un modelo para la detección o segmentación de objetos en imágenes, el índice de Jaccard está dado por:

$$\text{IoU} = \frac{|\hat{y}_1 \cap y_1|}{|\hat{y}_1 \cup y_1|}$$

Formalmente, IoU : $M\left(\mathbb{X}|\theta\right) \times \mathbb{Y} \rightarrow [0, 1]$, donde IoU=0 indica un resultado deficiente y IoU=1 indica una coincidencia exacta.

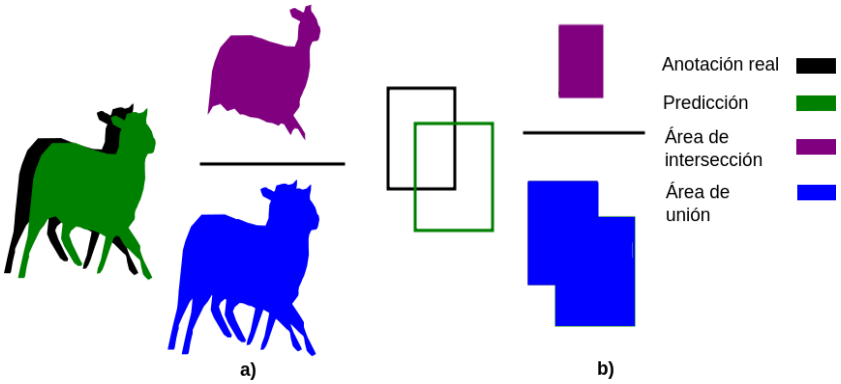


Figura 2.8: Índice Jaccard.

Representación gráfica del índice Jaccard. a) Muestra la ilustración del índice para el caso de una segmentación. b) Muestra la ilustración del índice para el caso de una detección.

El **promedio de precisión**, también denotado como **mAP** (*mean average precision*), es una métrica utilizada para evaluar el rendimiento de modelos en tareas de detección de objetos, especialmente en problemas en los que se detectan múltiples objetos en una imagen.

El mAP combina la precisión y la sensibilidad, pero de una manera más detallada. Para cada clase, el modelo calcula la precisión en diferentes niveles de confianza,

creando una curva (Precision-sensibilidad). El área bajo esta curva se llama Average Precision (AP). El mAP simplemente promedia los valores de AP para todas las clases.

Para calcular esta métrica, se procede de la siguiente manera:

- **Cálculo de la Precisión y la sensibilidad por Clase:** Para cada clase, se calcula la precisión y el recall en diferentes umbrales de confianza.
- **Construcción de la Curva Precisión-Sensibilidad:** Para cada clase, se construye una curva que representa la relación entre la precisión y la sensibilidad.
- **Cálculo del Área Bajo la Curva PR:** Se calcula el área bajo la curva para cada clase.
- **Promedio de las Áreas Bajo la Curva:** Se promedian las áreas bajo la curva para todas las clases para obtener el mAP.

Formalmente, Sea C el conjunto de clases y $AP(c)$ el promedio de precisión para una clase c. Entonces, el mAP se define como:

$$mAP = \frac{1}{|C|} \sum_{c \in C} AP(c)$$

Donde $AP(c)$ es el área bajo la curva de Precisión-sensibilidad para la clase c:

$$AP(c) = \int_0^1 P(r)\, dr$$

Aquí:

- $P(r)$: Precisión como función del *sensibilidad* r.
- r: *sensibilidad*, definido como la proporción de verdaderos positivos encontrados respecto al total de positivos reales.
- La integral se aproxima generalmente sumando puntos discretos en la curva *Precision-sensibilidad*.

Finalmente, en el presente trabajo se ocupa extensamente el mAP50 el cual se refiere al cálculo del mAP usando un umbral de IoU fijo del 50 %. Esto significa que para que una detección sea considerada correcta, al menos el 50 % del área de la predicción debe coincidir con el área real del objeto. Es una métrica común en evaluación porque da una idea básica de qué tan bueno es un modelo detectando objetos sin ser demasiado estricto.

En resumen:

El mAP50 mide el desempeño del modelo con un criterio moderado (IoU $> 50\,\%$). En contraste, valores como mAP$[50 : 95]$ evalúan el modelo en un rango más amplio de IoU (del 50 % al 95 %).

Capítulo 3

Redes neuronales

En este capítulo se exploran los fundamentos teóricos y conceptuales necesarios para entender las redes neuronales y su aplicación en la identificación y localización de objetos.

3.1. Redes neuronales artificiales

Las redes neuronales artificiales son modelos computacionales inspirados en el funcionamiento del cerebro humano, específicamente en la forma en que las neuronas interactúan en la red neuronal biológica. Estas redes son una parte fundamental del campo de la inteligencia artificial y el aprendizaje profundo.

3.1.1. Conceptos de redes neuronales

Una **red neuronal artificial** es un modelo constituido por capas, donde cada capa se compone de un número determinado de nodos (neuronas), que son representaciones gráficas de variables numéricas como se ilustra en la figura 3.1. El modelo comienza con una capa de entrada, seguida de capas ocultas y finalizando con la capa de salida. Los valores almacenados en la capa n se utilizan para calcular los valores almacenados en los nodos de la capa número $n + 1$, a través de un proceso de entrenamiento que permite la actualización de los valores almacenados en la red mediante un proceso de optimización recursiva.

Este sistema es análogo a las redes biológicas de neuronas, donde una neurona transmite una señal eléctrica (activación) mediante un proceso llamado sinapsis, que permite propagar la señal eléctrica a las neuronas cercanas y, por ende, propagar la señal por toda la red.

Para información detallada sobre neuronas biológicas, véase [56].

Una **neurona artifcial** es la unidad básica de procesamiento de una red neuronal, esta modela de manera simplificada el comportamiento de las neuronas biológicas presentes en el cerebro humano. Cada neurona artificial realiza operaciones matemáticas en las entradas que recibe, produce una salida y contribuye al proceso de aprendizaje de la red así mismo es representada gráficamente como un nodo que caracterizado por su entrada, peso, sesgo, función de activación y salida.

Una **conexión** se define como la relación entre las neuronas de diferentes capas de la red, estas permiten la propagación de la información dentro de la red y se repre-

sentan gráficamente como las aristas de la red. Las conexiones son parte esencial del diseño de una red neuronal ya que permite la capacidad para aprender patrones complejos en los datos de entrada y propagar la informacion dentro de la red.

Así los **pesos** son los parámetros ajustables que la red neuronal utiliza para aprender o aproximar funciones a partir de los datos de entrada durante el proceso de entrenamiento. Cada conexión entre dos neuronas tiene asociado un peso que indica la influencia de la salida de una neurona vista como entrada de la siguiente neurona.

Los **sesgos** son parámetros adicional asociado a cada neurona, aparte de los pesos. El sesgo proporciona a la red neuronal cierta flexibilidad y la capacidad de aprender patrones incluso cuando todos los pesos son inicialmente nulos o aleatorios.

La **función de activación** es una función matemática que transforma la entrada ponderada de la neurona artificial. Estas funciones permiten a la red aproximar funciones complejas a traves de la combinación ponderada de funciones de activacion.

De esta forma una **capa** se define como un conjunto de neuronas o unidades de procesamiento organizadas de manera conjunta en una estructura específica. Estas capas son los bloques fundamentales que componen una red neuronal y son responsables de realizar transformaciones en las entradas de la red, que permiten el aprendizaje o aproximación de funciones complejas.

Toda red neuronal funcional cuenta con tres tipos básicos de capas

- **Capa de Entrada:** Esta capa recibe las entradas del modelo, que pueden ser píxeles de una imagen,seires de tiempo u otro tipo de datos. Cada neurona en esta capa esta asignada una característica específica.
- **Capa de Salida:** Esta capa produce las predicciones o resultados finales del modelo. La cantidad de neuronas en esta capa depende de la naturaleza de cada problema.
- **Capas Ocultas:** Son capas intermedias entre la capa de entrada y la capa de salida. Cada neurona en estas capas realiza operaciones matemáticas en sus entradas y contribuye a la representación y transformación de los datos. La presencia de una gran cantidad de capas ocultas hace que la red sea "profunda".

Una **función de pérdida** o **costo** mide la discrepancia entre las predicciones del modelo y las etiquetas en el conjunto de datos. Cuanto menor sea el valor de la función de pérdida, mejor será la calidad de las predicciones del modelo. Estas funciones son utilizadas durante cada iteración del entrenamiento para calcular la pérdida y ajustar los parámetros del modelo mediante algoritmos de optimización que buscan valores minimos de dicha función.

Los **parámetros** del modelo son las variables internas ajustables que el modelo utiliza para aproximar y realizar predicciones o inferencias a partir de los datos de entrada. Estos parámetros son los valores númericos que la red neuronal o el modelo de machine learning ajusta(aprende) durante el proceso de entrenamiento,

por ejemplo, los pesos y los sesgos.

Por otro lado un **hiperparámetro** es un valor externo a un modelo de aprendizaje automático que no se aprenden durante el proceso de entrenamiento, sino que se establecen a modo de configuuración antes de iniciar el entrenamiento. Estos afectan el comportamiento y rendimiento del modelo; es decir, los hiperparámetros se eligen de antemano y generalmente se ajustan mediante prueba y error, búsqueda en cuadrícula u otras técnicas avanzadas.

Formalmente, si se tienen k valores almacenados en las neuronas de la n-ésima capa de una red neuronal, a cada neurona en esta capa se le asigna un peso para cada nodo en la $(n+1)$-ésima capa. Así, $w^n_{a,b} \in \mathbb{R}$ representa el peso de la a-ésima neurona en la n-ésima capa ligado a la conexión con la b-ésima neurona de la $(n+1)$-ésima capa. De esta manera, podemos representar los pesos entre dos capas de una red neuronal de forma matricial

$$W_n = \begin{bmatrix} w^n_{1,1} & w^n_{1,2} & \cdots & w^n_{1,k} \\ w^n_{2,1} & w^n_{2,2} & \cdots & w^n_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ w^n_{j,1} & w^n_{j,2} & \cdots & w^n_{j,k} \end{bmatrix}$$

donde W_n es la matriz de pesos que conecta la capa n-ésima con la $(n+1)$-ésima capa.

Agregando el sesgo definido por una constante $b \in \mathbb{R}^j$ y la función de activación $\sigma : \mathbb{R} \to \mathbb{R}$, utilizada para convertir el valor calculado por los pesos y sesgos en uno nuevo que se almacena en las neuronas de la red.

Tenemos que, para calcular el estado de la capa n-ésima, se tiene la siguiente expresión recursiva

$$A_n = \sigma(W_n A_{n-1} + b_n) = \sigma\left(\begin{bmatrix} w^n_{1,1} & w^n_{1,2} & \cdots & w^n_{1,k} \\ w^n_{2,1} & w^n_{2,2} & \cdots & w^n_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ w^n_{j,1} & w^n_{j,2} & \cdots & w^n_{j,k} \end{bmatrix} \begin{bmatrix} a^{n-1}_1 \\ a^{n-1}_2 \\ \vdots \\ a^{n-1}_k \end{bmatrix} + \begin{bmatrix} b^n_1 \\ b^n_2 \\ \vdots \\ b^n_j \end{bmatrix} \right)$$

donde

$$\sigma\left(\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,k} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ a_{j,1} & a_{j,2} & \cdots & a_{j,k} \end{bmatrix} \right) = \begin{bmatrix} \sigma(a_{1,1}) & \sigma(a_{1,2}) & \cdots & \sigma(a_{1,k}) \\ \sigma(a_{2,1}) & \sigma(a_{2,2}) & \cdots & \sigma(a_{2,k}) \\ \vdots & \vdots & \ddots & \vdots \\ \sigma(a_{j,1}) & \sigma(a_{j,2}) & \cdots & \sigma(a_{j,k}) \end{bmatrix}.$$

Así, la función final calculada por una red de profundidad N está dada por la siguiente composición

$$F(x) = \sigma\left(W_N \sigma\left(\ldots \sigma\left(W_2 \sigma(W_1 x + b_1) + b_2\right)\ldots\right) + b_N\right).$$

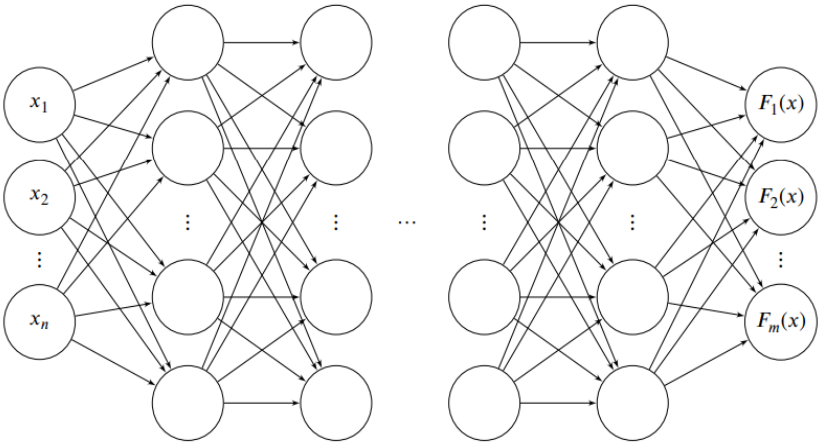


Figura 3.1: Red neuronal artificial.
Representación de una arquitectura básica de una red neuronal artificial.

3.1.2. Funciones de activación

La elección particular de las funciones de activación para las neuronas de una capa específica depende de las características del problema a resolver, la experiencia y el nivel teórico; no hay una metodología universal para su elección. En esta sección, se mencionan las funciones de activación que aplican los modelos que se utilizaran en el desarrollo del presente trabajo.

- **Logística, sigmoide o escalón suave**

$$\sigma(x) = \frac{1}{1+e^x}$$

- **ReLU (Unidad linealmente rectificada)**

$$\sigma(x) = \begin{cases} 0 & si \quad x \leqslant 0 \\ \\ x & si \quad x > 0 \end{cases}$$

- **Leaky-ReLU (Unidad linealmente rectificada modificada)**

$$\sigma(x) = \begin{cases} 0{,}01x & si \quad x < 0 \\ \\ x & si \quad x \geqslant 0 \end{cases}$$

- **Softmax (máximo suave)**

$$\sigma\left(\overline{x}\right) = \frac{e^{x_k}}{\sum_n^N e^{x_n}} \text{ donde } |\overline{x}| = N$$

Para información detallada sobre funciones de activación, véase [51].

3.1.3. Funciones de pérdida

Las funciones de pérdida proporcionan una medida que cuantifica el aprendizaje y, por ende, la capacidad de predicción del modelo en función de la diferencia entre

las predicciones del modelo y las anotaciones del conjunto de datos.

Formalmente, suponiendo que se desea aproximar una función $f : \mathbb{R}^n \to \mathbb{R}^m$, y que una red $M(x|\theta)$ define una aproximación mediante una función $F_i : \mathbb{R}^n \to \mathbb{R}^m$.

Para cuantificar la calidad de la predicción dada por el modelo, se establece una función de pérdida que mide la discrepancia entre dos funciones. Dado un conjunto de validación o prueba con N datos $(x_i, y_i) \in \mathbb{X} \times \mathbb{Y}$, que son descritos por la relación $f(x_i) = y_i$, estos pueden ser proporcionados a la red para obtener predicciones $F_i(x_i) = \hat{y}_i$ de manera que la función de costo $C : \mathbb{X} \times M(\mathbb{X}|\theta) \to \mathbb{R}$ promedia los errores, brindando una estimación de la capacidad predictiva del modelo.

Al igual que con las funciones de activación, existen distintas funciones de pérdida y la elección de cuál utilizar depende de las características del problema a resolver, la experiencia y el nivel teórico del desarrollador. En esta parte, se mencionan la funciones utilizadas por los modelos que implementados en el desarrollo del presente trabajo.

Sean $\vec{y} = \{y_1, \cdots, y_n\}$ el conjunto de etiquetas de prueba y $\hat{\vec{y}} = \{\hat{y}_1, \cdots, \hat{y}_n\}$ el conjunto de predicciones hechas por el modelo para los datos de prueba. Se tienen las siguientes funciones de pérdida:

- **Raíz cuadrada media (RMSE)**
  Se entiende como residuos dados por la diferencia entre el valor predicho y el valor real obtenido.
  Características
  - Penaliza los valores que son muy grandes.
  - No es fácil de interpretar.
  - Funciona bien para optimizar regresiones en general.

  $$C(\vec{y}, \hat{\vec{y}}) = \sqrt{\sum_{i=1}^{n} \frac{(\hat{y}_i - y_i)^2}{n}}$$

  Para información detallada sobre RMSE, véase [53].

- **Entropía cruzada categórica Categorical (Cross-Entropy)**
  La entropía cruzada categórica, es una medida de precisión para variables categóricas.
  Características
  - Difícil de derivar y de que converja.
  - Escala univariante.
  - Simétrica.
  - Es fácil de interpretar.

  $$C(\vec{y}, \hat{\vec{y}}) = -\frac{1}{n} \sum_{k=1}^{n} \sum_{i=1}^{c} y_i \log(\hat{y}_i)$$

  Donde:

- i es la clase.
- c el número de clases.
- y_i la clase real.
- $\hat{y}_i$ la clase predicha.

Para información detallada sobre entropía cuzada, véase [55].

3.1.4. Optimizadores

Dada la interpretación de aproximadores de funciones, una red neuronal $M(x|\theta)$ debe ajustar sus predicciones de manera que garantice matemáticamente la mejora en la convergencia a la función de interés. Esto se logra a través de algoritmos que buscan minimizar la función de costo utilizada en el entrenamiento del modelo.

Estos algoritmos reciben el nombre de optimizadores y están basados en conceptos de cálculo multivariable. La idea general es la siguiente.

Dada una función diferenciable $f : \mathbb{R}^n \to \mathbb{R}$, definimos el gradiente $\nabla f : \mathbb{R}^n \to \mathbb{R}$ evaluado en un punto $\mathbf{x} \in \mathbb{R}^n$ mediante $\nabla f(\mathbf{x}) = \left(\frac{\partial f}{\partial x_1}(\mathbf{x}), \ldots, \frac{\partial f}{\partial x_n}(\mathbf{x})\right)$, el cual indica la dirección del ascenso más pronunciado descrito por la función. Dado que en el punto $\mathbf{x}$ la función es continua y diferenciable, puede obtenerse una linearización; por lo tanto, el vector $-\nabla f(\mathbf{x})$ indica la dirección del descenso más pronunciado descrito por la función.

Para obtener el valor mínimo de la función, se debe comenzar en un punto inicial $x_0 \in \mathbb{R}^n$, calcular el valor $-\nabla f(x_0)$ y luego proceder a calcular un nuevo punto $x_1 = x_0 - \lambda \nabla f(x_0)$, donde $\lambda \geqslant 0$ se denomina tasa de aprendizaje.

Este proceso continúa iterativamente hasta que nos acercamos a una región cercana a nuestro mínimo deseado, proporcionando así los parámetros adecuados para el modelo.

Los optimizadores ofrecen distintas estrategias de aproximación a los mínimos, todos basados en esta idea general. En esta sección, se enlistan algunos de los más populares. La elección de cuál utilizar depende del problema y la naturaleza de la función a minimizar.

Descenso estocástico de gradiente (SGD)

El término "estocástico" se refiere a que este algoritmo utiliza una muestra aleatoria de ejemplos de entrenamiento en cada iteración para calcular la dirección del descenso del gradiente.
Dada una función de costo $C(\theta)$, donde θ representa los parámetros del modelo, SGD actualiza los parámetros mediante la regla

$$\theta_{t+1} = \theta_t - \alpha \nabla C(\theta_t)$$

Aquí, α es el hiperparámetro de la tasa de aprendizaje, que controla el tamaño del paso que se toma en la dirección opuesta al gradiente. El gradiente $\nabla C(\theta_t)$ indica la dirección y magnitud del mayor incremento en la función de costo en el punto θ_t. Al restar este gradiente, ponderado por la tasa de aprendizaje, se realiza un

descenso en la dirección que minimiza la función de costo.

En cada iteración de SGD, se selecciona un subconjunto aleatorio de muestras del conjunto de entrenamiento en lugar de usar todo el conjunto de entrenamiento completo (descenso de gradiente clásico). Esta aproximación estocástica hace que el algoritmo sea computacionalmente más eficiente y más adecuado para conjuntos de datos grandes.

El proceso de actualización de los parámetros mediante SGD se repite hasta que se cumpla un criterio de parada, como un número máximo de iteraciones o cuando la mejora en la función de costo es menor que un umbral predefinido.

Si bien SGD es efectivo, también presenta algunos desafíos. Debido a la naturaleza estocástica de la selección de lotes, el algoritmo puede converger más lentamente y puede quedar atrapado en óptimos locales. Para abordar estos problemas, han surgido variantes de SGD, como Momentum, Adam y RMSprop, que ajustan la dirección y el tamaño de los pasos de actualización para mejorar el rendimiento del algoritmo.

Para obtener información detallada sobre SGD, véase [57].

Descenso estocástico de gradiente con impulso (Momentum)

Es una variante del SGD que mejora la convergencia y suaviza el proceso de actualización de los parámetros durante el entrenamiento de redes neuronales.

El objetivo principal es minimizar una función de costo ajustando los parámetros del modelo. A diferencia del SGD estándar, este algoritmo utiliza un promedio acumulado de los gradientes calculados en la iteración previa para guiar las actualizaciones de los parámetros, lo que le permite tener en cuenta la historia de los gradientes.

La actualización de los parámetros en el SGD con Momentum se realiza mediante las siguientes fórmulas

$$v_{t+1} = \mu v_t - \alpha \nabla C(\theta_t)$$
$$\theta_{t+1} = \theta_t + v_{t+1}$$

Aquí, α es el hiperparámetro de la tasa de aprendizaje, que determina el tamaño del paso en la dirección opuesta al gradiente, y μ es el coeficiente de momentum, que controla la contribución de los gradientes pasados en la actualización. El término $\nabla C(\theta_t)$ representa el gradiente de la función de costo en el punto θ_t.

El algoritmo comienza inicializando v_0 como un vector de ceros. En cada iteración, se calcula el promedio ponderado de los gradientes anteriores mediante μv_t, lo que introduce un impulso que mantiene la dirección de la actualización. Luego, se calcula el gradiente actual $\nabla C(\theta_t)$ y se realiza la actualización de los parámetros θ_{t+1} sumando el promedio acumulado v_{t+1}.

El uso de momentum en el algoritmo ayuda a suavizar las oscilaciones y el ruido inherentes al descenso del gradiente, lo que puede acelerar la convergencia. Además, el momentum también puede ayudar a escapar de óptimos locales y superar

regiones de gradientes planos.

Es importante ajustar adecuadamente la tasa de aprendizaje α y el coeficiente de momentum μ para obtener un buen rendimiento. Un valor de μ cercano a 1 conserva en mayor medida la información de los gradientes pasados, mientras que un valor bajo de μ reduce esta influencia.

Para obtener información detallada sobre este método, véase [58].

Algoritmo de gradiente adaptativo (AdaGrad)

Se caracteriza por adaptar la tasa de aprendizaje de forma automática para cada parámetro del modelo. Este algoritmo ajusta la tasa de aprendizaje según la magnitud de los gradientes anteriores para acelerar la convergencia en dimensiones con gradientes que tienden a cero.

Se utiliza un conjunto de gradientes acumulados para adaptar la tasa de aprendizaje en cada paso de actualización de los parámetros. La actualización de los parámetros se realiza utilizando las siguientes fórmulas

$$g_t = \nabla C(\theta_t)$$
$$G_t = G_{t-1} + g_t^2$$
$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{G_t + \varepsilon}} * g_t$$

Aquí, g_t es el gradiente de la función de costo C en el paso t, G_t es una matriz diagonal que contiene la suma acumulada de los gradientes al cuadrado hasta el paso t, θ_t representa los parámetros del modelo en el paso t, α es la tasa de aprendizaje inicial y ε es una pequeña constante para evitar divisiones por cero.

La idea principal es que las dimensiones con gradientes grandes suelen requerir pasos de aprendizaje más pequeños, mientras que las dimensiones con gradientes pequeños pueden permitirse pasos de aprendizaje más grandes.

El algoritmo logra esto adaptando la tasa de aprendizaje para cada dimensión en función de la historia acumulada de los gradientes.

Una de las ventajas es que no requiere ajustes manuales de la tasa de aprendizaje, ya que esta se adapta automáticamente durante el entrenamiento. Sin embargo, una limitación es que la acumulación de los gradientes al cuadrado en la matriz G_t puede hacer que la tasa de aprendizaje se vuelva demasiado pequeña a medida que aumentan las iteraciones, lo que puede frenar el proceso de aprendizaje.

Es un algoritmo popular en el entrenamiento de redes neuronales, especialmente en problemas donde los datos son escasos o donde las características tienen diferentes escalas. Proporciona una adaptación eficiente de la tasa de aprendizaje y puede ayudar a mejorar la convergencia y el rendimiento del modelo.

Para obtener información detallada sobre el método, véase [59].

Propagación de raíz cuadrática media (RMSProp)

Este algoritmo ajusta la tasa de aprendizaje según la magnitud de los gradientes recientes para mejorar la convergencia en problemas con gradientes escasos o variables.

Se utiliza la media de los cuadrados de los gradientes para adaptar la tasa de aprendizaje en cada paso de actualización de los parámetros. La actualización de los parámetros se realiza utilizando las siguientes fórmulas

$$g_t = \nabla C(\theta_t)$$
$$E[g^2]_t = \beta E[g^2]_{t-1} + (1-\beta) g_t^2$$
$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{E[g^2]_t + \varepsilon}} * g_t$$

Aquí, g_t es el gradiente de la función de costo C en el paso t, $E[g^2]_t$ es una media de los cuadrados de los gradientes hasta el paso t, θ_t representa los parámetros del modelo en el paso t, α es la tasa de aprendizaje inicial, β es el factor de decaimiento y ε es una pequeña constante para evitar divisiones por cero.

La idea principal es ajustar la tasa de aprendizaje para cada parámetro según la magnitud de los gradientes recientes. Al utilizar la media de los cuadrados de los gradientes, se da mayor importancia a los gradientes más recientes y se reduce la influencia de los gradientes pasados. Esto permite que la tasa de aprendizaje se adapte de forma más sensible a las variaciones en la topología de la función de costo.

Proporciona una adaptación eficiente de la tasa de aprendizaje y puede mejorar el rendimiento del modelo en problemas con gradientes variables o escasos.

Para información detallada sobre el método, véase [60].

Optimización de momento adaptativo (Adam)

Este algoritmo combina las ventajas del algoritmo RMSProp y el Momentum. Adam adapta la tasa de aprendizaje de forma automática para cada parámetro del modelo y mantiene una estimación de los momentos de primer y segundo orden de los gradientes.

Se utilizan dos momentos para adaptar la tasa de aprendizaje en cada paso de actualización de los parámetros. Los momentos de primer y segundo orden se calculan utilizando las siguientes fórmulas:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

Donde g_t es el gradiente de la función de costo C en el paso t, m_t es el momento de primer orden y v_t es el momento de segundo orden.

Para corregir el sesgo de los momentos en las primeras iteraciones, se utilizan los siguientes pasos de corrección de sesgo:

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

Luego, los parámetros se actualizan utilizando la siguiente fórmula:

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{\hat{v}_t} + \varepsilon}\hat{m}_t$$

Aquí, θ_t representa los parámetros del modelo en el paso t, α es la tasa de aprendizaje inicial, β_1 y β_2 son factores de decaimiento, y ε es una pequeña constante para evitar divisiones por cero.

Combina el uso de momentos de primer y segundo orden para adaptar la tasa de aprendizaje de forma eficiente. Los momentos de primer orden ayudan a estabilizar la dirección de las actualizaciones, mientras que los momentos de segundo orden escalan la tasa de aprendizaje de acuerdo con la magnitud de los gradientes.

Su capacidad para adaptar la tasa de aprendizaje y mantener estimaciones de los momentos de primer y segundo orden lo convierte en un algoritmo robusto para una variedad de problemas.

Para información detallada sobre el método, véase [60].

AdaDelta

Es un algoritmo que adapta la tasa de aprendizaje de forma automática para cada parámetro del modelo. A diferencia de otros algoritmos, no utiliza una tasa de aprendizaje fija, sino que ajusta la tasa de aprendizaje según el historial de actualizaciones anteriores.

Se utiliza la media de los cuadrados de las actualizaciones anteriores para adaptar la tasa de aprendizaje en cada paso de actualización de los parámetros. La actualización de los parámetros se realiza utilizando las siguientes fórmulas:

$$g_t = \nabla C(\theta_t)$$
$$E[g^2]_t = \rho E[g^2]_{t-1} + (1-\rho) g_t^2$$
$$\Delta\theta_t = -\frac{\sqrt{\Delta\theta_{t-1} + \epsilon}}{\sqrt{E[g^2]_t + \varepsilon}} * g_t$$
$$\theta_{t+1} = \theta_t + \Delta\theta_t$$
$$E[\Delta\theta^2]_t = \rho E[\Delta\theta^2]_{t-1} + (1-\rho)\Delta\theta_t^2$$

Aquí, g_t es el gradiente de la función de costo C en el paso t, $E[g^2]_t$ es la media de los cuadrados de los gradientes hasta el paso t, θ_t representa los parámetros del modelo en el paso t, $\Delta\theta_t$ es la actualización de los parámetros en el paso t, ε es una pequeña constante para evitar divisiones por cero y ρ es un factor de decaimiento.

La idea principal es que la tasa de aprendizaje se adapte según el historial de actualizaciones anteriores. En lugar de utilizar una tasa de aprendizaje fija, utiliza la relación entre las actualizaciones recientes y los gradientes pasados para ajustar la tasa de aprendizaje de forma adaptativa.

Una de las ventajas es que elimina la necesidad de ajustar manualmente la tasa de aprendizaje, ya que esta se adapta automáticamente durante el entrenamiento.

Además, puede ayudar a superar los problemas de convergencia lenta o explosiva que pueden ocurrir con tasas de aprendizaje fijas.

Para información detallada sobre el método, véase [62].

3.1.5. Retropropagación

Al trabajar con aproximaciones de funciones, no solo se requiere que las predicciones del modelo sean óptimas. Si la función de costo muestra una diferencia considerable entre las predicciones y el comportamiento deseado de la red, es necesario reajustar los parámetros θ del modelo $M(x|\theta)$. Esto se logra en el proceso de entrenamiento mediante el método conocido como **retropropagación**, una técnica de optimización que permite que la red ajuste sus parámetros en función del error cometido en las predicciones, con el objetivo de mejorar el rendimiento del modelo.

El principal desafío de aplicar el descenso de gradiente a las redes neuronales es calcular las derivadas parciales de la función de costo con respecto a cada peso y sesgo individual, es decir, $\dfrac{\partial C}{\partial w_{i,j}}$ y $\dfrac{\partial C}{\partial b_j}$.

Aquí es donde entra la retropropagación. Este algoritmo nos ayuda a calcular los valores para la última capa de conexiones y, con estos resultados, avanza de manera inductiva hacia atrás a través de la red, calculando las derivadas parciales de cada capa hasta llegar a la primera. De ahí su nombre.

Por cuestiones de simplicidad, se considera la función de costo sobre un solo dato etiquetado x.

Dado que solo se sabe cómo calcular ∇C_x para un punto.
x, se utiliza la relación

$$\nabla C = \nabla \left(\sum_{i=1}^{N} C_{x_i} \right) = \sum_{i=1}^{N} \nabla C_{x_i}.$$

Lo que permite llevar a cabo este proceso para cada punto y sumar los valores del gradiente. Esto es importante porque utilizar la retropropagación en un conjunto de entrenamiento grande para cada iteración del entrenamiento se convierte en un enfoque computacionalmente costoso. En cambio, si se seleccionan algunos elementos del conjunto de entrenamiento para calcular el gradiente y se actualiza la red, se obtienen resultados satisfactorios a través de un proceso recursivo.

Sea $z_j^l = \sum_k w_{j,k}^l a_k^{l-1} + b_j^l$ donde $a_j^l = \sigma(z_j^l)$ y $A^l = \sigma(Z^l)$ representan los valores enviados por la capa $(n-1)$-ésima antes de aplicar la función de activación y $Z^l := \sum_k z_k^l e_k$ es un vector con entradas correspondientes a los valores z_j^l y e_k son vectores de la base canonica.

Considerando $\delta_j^l = \dfrac{\partial C}{\partial z_j^l}$ y $\Delta^l = \sum_k \delta_k^l e_k$, estos valores son útiles para propagar el algoritmo hacia atrás a través de la red y están directamente relacionados con $\dfrac{\partial C}{\partial w_{i,j}}$ y $\dfrac{\partial C}{\partial b_j}$ mediante la regla de la cadena :

$$\frac{\partial C}{\partial w_{i,j}^l} = \frac{\partial C}{\partial z_j^l}\frac{\partial z_j^l}{\partial w_{i,j}^l} = \delta_j^l a_i^{l-1} \text{ y } \frac{\partial C}{\partial b_j^l} = \frac{\partial C}{\partial z_j^l}\frac{\partial z_j^l}{\partial b_j^l} = \delta_j^l$$

Dado que a_j^{l-1} es un valor disponible para cualquier nodo de la red, si se calcula el valor de δ_j^l, se resuelve el problema de clacular el gradiente.

El primer paso es calcular este valor para la última capa de la red, es decir, δ_j^L, para una red con L capas. Dado que, nuevamente, por la regla de la cadena se tiene

$$\delta_j^L = \frac{\partial C}{\partial a_j^L}\frac{\partial a_j^L}{\partial z_j^L} = \frac{\partial C}{\partial a_j^L}\sigma'(z_j^L)$$

observando que $F(x) = A^L = (a_1^L, \ldots, a_k^L)$ y $f(x) = (y_1, \ldots, y_k)$ tenemos

$$\delta_j^L = (a_j^L - y_j)\sigma'(z_j^L)$$

Lo cual se puede calcular fácilmente mediante una computadora. Expresando este resultado como vector se tiene

$$\Delta^L = \nabla_{A^L} \odot \sigma'(Z^L).$$

Donde $\nabla_{A^L} = \left(\frac{\partial C}{\partial a_1^L}, \ldots, \frac{\partial C}{\partial a_k^L}\right)$ es el gradiente de C tomado con respecto a los elementos de A^L y $\odot$ es el Producto de Hadamard, definido como:

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,k} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ a_{j,1} & a_{j,2} & \cdots & a_{j,k} \end{bmatrix} \odot \begin{bmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,k} \\ b_{2,1} & b_{2,2} & \cdots & b_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ b_{j,1} & b_{j,2} & \cdots & b_{j,k} \end{bmatrix} = \begin{bmatrix} a_{1,1}b_{1,1} & a_{1,2}b_{1,2} & \cdots & a_{1,k}b_{1,k} \\ a_{2,1}b_{2,1} & a_{2,2}b_{2,2} & \cdots & a_{2,k}b_{2,k} \\ \vdots & \vdots & \ddots & \vdots \\ a_{j,1}b_{j,1} & a_{j,2}b_{j,2} & \cdots & a_{j,k}b_{j,k} \end{bmatrix}.$$

Ahora solo resta propagar este procedimiento hacia atrás en la red para obtener δ_j^{L-1}, esto se logra aplicando la regla de la cadena

$$\delta_j^{L-1} = \frac{\partial C}{\partial z_j^{L-1}} = \nabla_{Z^L} C \cdot \frac{\partial Z^L}{\partial z_j^{L-1}} = \sum_i^k \frac{\partial C}{\partial z_i^L}\frac{\partial z_i^L}{\partial z_j^{L-1}} = \sum_i^k \delta_i^L \frac{\partial z_i^L}{\partial z_j^{L-1}}.$$

Si nos concentramos en el termino $\dfrac{\partial z_i^L}{\partial z_j^{L-1}}$ tenemos

$$\frac{\partial z_i^L}{\partial z_j^{L-1}} = \frac{\partial}{\partial z_j^{L-1}}\left(\sum_k w_{i,k}^L a_k^{L-1} + b_i^L\right) = \frac{\partial}{\partial z_j^{L-1}}\left(\sum_k w_{i,k}^L \sigma(z_k^{L-1}) + b_i^L\right) =$$

$$\frac{\partial}{\partial z_j^{L-1}}\left(w_{i,j}^L \sigma(z_j^{L-1})\right) = w_{i,j}^L \sigma'(z_j^{L-1})$$

Lo cual, nuevamente, es fácil de calcular para una computadora. Por lo tanto

$$\delta_j^{L-1} = \sum_i^k \delta_i^L w_{i,j}^L \sigma'(z_j^{L-1})$$

Esta fórmula nos indica cómo calcular cualquier δ_j^l , conociendo Δ^{l+1}. Dado que sabemos cómo calcular Δ^L en la última capa de la red, tenemos completo el algoritmo de retropropagación.

3.1.6. Teoremas de aproximación.

Por mera formalidad matemática, enunciaremos y proporcionaremos un esbozo de la demostración de los siguiente teoremas, donde el primero nos garantiza que, dada una función de activación que cumple ciertas propiedades, puede ser utilizada por una red neuronal para aproximar funciones continuas. El segundo permite extender el resultado a funciones de activación que son continuas.

Teorema 3.1.1.
Sea f una función discriminatoria continua. Entonces, una red neuronal con f como función de activación es un aproximador universal. [63].

Demostración.
*Sea n un número natural. Decimos que una función de activación $f\colon \mathbb{R} \to \mathbb{R}$ es n-**discriminatoria** si la única medida signada de Borel μ tal que*

$$\int f(y \cdot x + \theta)\, d\mu(x) = 0 \quad \text{para todo } y \in \mathbb{R}^n \text{ y } \theta \in \mathbb{R}$$

es la medida cero.

Así, decimos que f es discriminatoria si es n-discriminatoria para cualquier $n \in \mathbb{N}$.

*Sabiendo que dado un espacio topológico Ω y una función $f : \mathbb{R} \to \mathbb{R}$, definimos a una red neuronal con función de activación f como un **aproximador universal** en Ω si se cumple que $\Sigma_n(f)$ es denso en $C(\Omega)$.*

Donde $C(\Omega) = \{f : \Omega \to \mathbb{R} | f \in C^1\}$, es el conjunto de funciones continuas de Ω a $\mathbb{R}$ y $\Sigma_n(f) = Gen\{f(y \cdot x + \theta) \mid y \in \mathbb{R}^n, \theta \in \mathbb{R}\}$, donde $y \cdot x$ representa el producto punto estándar en $\mathbb{R}^n$. El conjunto $\Sigma_n(f)$ consta de todas las funciones que pueden ser calculadas por una red neuronal con una sola capa oculta y función de activación f.

Buscando la contradicción, supongamos que $\Sigma_n(f)$ no es denso en $C(I_n)$ donde $I_n = [0,1]^n = \{x = (x_1, \ldots, x_n) \in \mathbb{R}^n \mid x_i \in [0,1] \text{ para cualquier } i = 1, \ldots, n\}$.
Por definición de densidad, se sigue que $\overline{\Sigma_n(f)} \neq C(I_n)$.

Luego, apelando a el teorema de Hahn-Banach que dice que dado un espacio vectorial normado $(V, ||\cdot||)$ y dos conjuntos $A, B \subseteq V$ no vacíos, cerrados, disjuntos y convexos, tal que uno de ellos es compacto. Entonces, existe un funcional lineal continuo $f \neq 0$, algún $\alpha \in \mathbb{R}$ y un $\varepsilon > 0$ tal que $f(x) \leq \alpha - \varepsilon$ para cualquier $x \in A$ y $f(y) \geq \alpha + \varepsilon$ para cualquier $y \in B$.

Esto nos garantiza que dado un espacio vectorial normado real $(V(\mathbb{R}), ||\cdot||)$ y $U \subseteq V(\mathbb{R})$ un subespacio lineal tal que $U \neq V$. Entonces, existe una aplicación lineal continua $f\colon V \to \mathbb{R}$ con $f(x) = 0$ para cualquier $x \in U$, y $f \not\equiv 0$.(*)

Dado que $I_n \subset \mathbb{R}$ cumple con las hipotesis anteriores, se puede concluir que existe algún funcional lineal continuo $F\colon C(I_n) \to \mathbb{R}$ tal que $F \neq 0$ pero $F(g) = 0$ para cualquier $g \in \Sigma_n(f)$.

Usando el Teorema de Representación de Riesz que nos dice, que dado Ω un subconjunto de $\mathbb{R}^n$ y $F : C(\Omega) \to \mathbb{R}$ un funcional lineal en el espacio de funciones

reales continuas con dominio en Ω. Entonces, existe una medida Borel con signo μ en Ω tal que para cualquier $f \in C(\Omega)$, tenemos que

$$F(f) = \int_{\Omega} f(x)\, d\mu(x).$$

garantizamos que existe alguna medida de Borel μ tal que

$$F(g) = \int_{I_n} g(x)\, d\mu(x) \text{ para todo } g \in C(I_n). \ (*)$$

Sin embargo, dado que para cualquier y y θ la función $f(y \cdot x + \theta)$ es un elemento de $\Sigma_n(f)$, esto significa que para todo $y \in \mathbb{R}^n$, y $\theta \in \mathbb{R}$ tenemos $\int f(y \cdot x + \theta)\, d\mu(x) = 0$, lo que significa que $\mu = 0$ (ya que f es discriminatorio) y, por lo tanto, $F(g) = 0$ para cualquier $g \in C(I_n)$.

Lo cual contradice claramente la afirmación (*) por lo que se debe tener que $\overline{\Sigma_n(f)} = C(I_n)$, que por definición grantiza la densidad, con esto concluye la prueba. ■

Teorema 3.1.2.
Aproximación universal
Una red neuronal artificial con una función de activación $f : \mathbb{R} \to \mathbb{R}$ continua, es un aproximador universal.

Demostración.
Tomando una función $f : \mathbb{R} \to \mathbb{R}$, tal que $f \in C^1$ por el teorema anterior basta demostar que si $\Sigma_1(f)$ es denso en $C([0,1])$, entonces $\Sigma_n(f)$ es denso en $C([0,1]^n$.

Para ello ocupemos el hecho de que el espacio generado por el conjunto $A = \{g(a \cdot x) \mid a \in \mathbb{R}^n, g \in C([0,1])\}$ es denso en $C([0,1]^n)$.

Esto significa que dada una funcion $h \in C([0,1]^n)$, existe una función en A tal que la distancia entre estas dos es minima.

Formalmente sea $h \in C([0,1]^n)$ y $\varepsilon > 0$, existe una función $g_k \in C([0,1])$ tal que

$$\left| h(x) - \sum_{k=1}^{N} g_k(a_k \cdot x) \right| < \frac{\varepsilon}{2}$$

Para cada función $g_k(a_k \cdot x)$ como se tiene la hipotesis de que $\Sigma_1(f)$ es denso en $C([0,1])$, se concluye que para toda g_k , existe una suma de funciones tal que cumple con

$$\left| g_k(a_k \cdot x) - \sum_{i=1}^{N_k} f(y_{k,i} \cdot x + \theta_{k,i}) \right| < \frac{\varepsilon}{2k}.$$

Aplicando la desigualdad del triangulo $\left| \sum_{i=0}^{N} x_i \right| \leqslant \sum_{i=0}^{N} |x_i|$ tenemos

$$\left| h(x) - \sum_{k=1}^{N} \sum_{i=1}^{N_k} f(y_{k,i} \cdot x + \theta_{k,i}) \right| < \left| h(x) - \sum_{k=1}^{N} g_k(a_k \cdot x) \right| + \frac{k(\varepsilon/2k)}{2k} < \frac{\varepsilon}{2} + \frac{\varepsilon}{2} = \varepsilon.$$

Esto demuetra que podemos aproximarnos arbitrariamente a cualquier función en $C([0,1]^n)$ usando funciones de $\Sigma_n(f)$, con lo que concluye la demostración.■

Para más detalles sobre los conceptos de análisis funcional usados en este teorema, véase [50].

3.1.7. Transferencia de conocimiento

El término se refiere a distintas técnicas que permiten utilizar el conocimiento de redes neuronales previamente entrenadas en una tarea específica para mejorar el rendimiento en una tarea relacionada o diferente.

Este enfoque es especialmente útil cuando existen problemas como conjuntos de datos limitados o cuando ya se ha entrenado un modelo en una tarea similar a la que se quiere realizar. Las técnicas más usadas se conforman por

- **Transferencia de Características:** Consiste en la reutilización de las representaciones aprendidas por un modelo en una tarea específica para otra tarea relacionada.
- **Transferencia de Modelo:** Implica reutilizar un modelo preentrenado en una tarea similar para inicializar los pesos de un modelo en una nueva tarea. Este enfoque puede acelerar el proceso de entrenamiento y mejorar el rendimiento.
- **Transferencia de Aprendizaje:** Se refiere a la transferencia de conocimiento desde una tarea fuente a una tarea objetivo. El modelo se entrena inicialmente en una tarea fuente y luego se adapta o ajusta a la tarea objetivo. Este enfoque es particularmente útil cuando las tareas comparten similitudes en la estructura de los datos o en los patrones subyacentes.
- **Transferencia de Conjunto de Datos:** Implica transferir el conocimiento al intercambiar o combinar conjuntos de datos entre tareas relacionadas. Esto puede ser útil cuando se tiene acceso a datos más ricos o abundantes en una tarea y se utilizan para mejorar el rendimiento en otra tarea.

De manera general, para aplicar la mayoría de las técnicas de transferencia de conocimiento, es necesario contar con un modelo preentrenado $M_1(x|\theta) \approx f$, el cual exhibe un rendimiento óptimo al desempeñar la tarea, caracterizada por una precisa aproximación a la función f. Gráficamente, se ilustra en la figura 3.2.

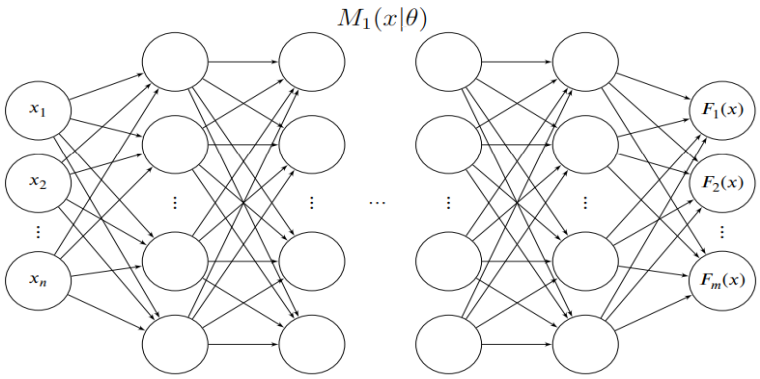


Figura 3.2: Red neuronal preentrenada.
Representación gráfica de una red neuronal artificial preentrenada $M_1(x|\theta)$.

Si la red preentrenada cuenta con N capas, el siguiente paso consiste en sustituir la primera y la N-ésima capa, es decir, la de entrada y la de salida, por aquellas que

se adecuen a las estructuras de los nuevos datos $\{(x'_1, y'_1), \cdots, (x'_n, y'_n)\} \subseteq \mathbb{X} \times \mathbb{Y}$ que vamos a utilizar para reentrenar la red, como se ilustra en la figura 3.3.

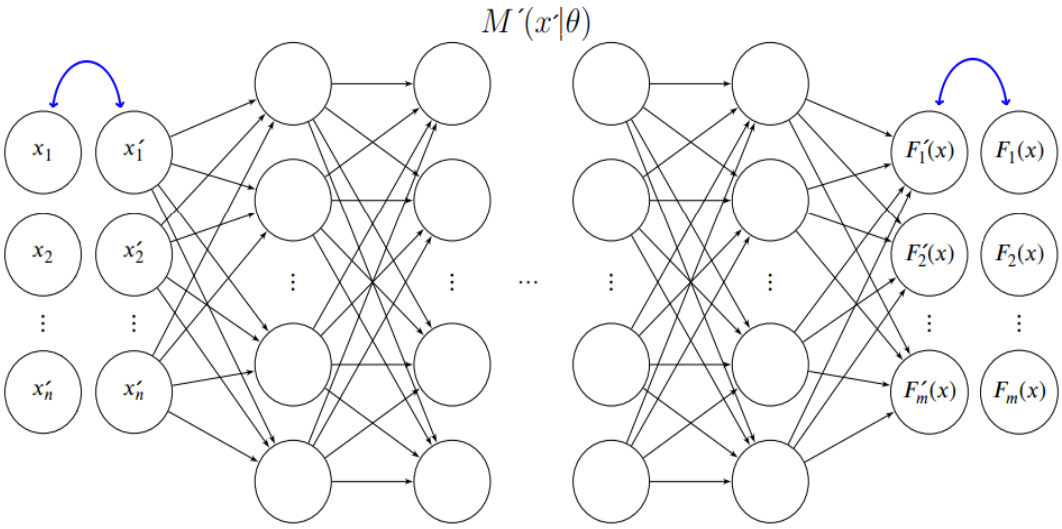


Figura 3.3: Red neuronal adecuada.

Representación gráfica de una red neuronal artificial preentrenada $M_1(x|\theta)$ a la que se le cambió la capa de salida y la de entrada para generar la nueva red $M'(x'|\theta)$.

Finalmente, se elige la técnica de transferencia de conocimiento a utilizar, lo que se traduce en seleccionar los parámetros $\theta' \subseteq \theta$ que se ocuparán para inicializar el modelo y definir cuáles serán reajustados en el reentrenamiento y cuáles se mantendrán fijos o congelados. Esto implica un reentrenamiento para el modelo $M'(x'|\theta')$, que dará paso al modelo final con transferencia de conocimiento $M_{\text{tf}}(x|\theta)$.

3.2. Aprendizaje profundo para detección de objetos

En esta sección se hace referencia a algunas de las herramientas de aprendizaje profundo en el ámbito de visión computacional desarrolladas para la detección de objetos y clasificación de imágenes.

3.2.1. Redes neuronales convolucionales

Conforman una de las principales herramientas para la visión computacional. Su nombre proviene del hecho de que cuentan con capas que realizan una operación matemática llamada **convolución**. De manera general, permiten extraer características de una imagen.

Cada píxel que conforma la imagen sirve de entrada a una neurona específica de la red. De esta forma, la red puede aprender características y relaciones de interés entre los píxeles de la imagen.

El problema esencial de una red neuronal no convolucional surge al analizar imágenes cuando se necesita clasificar dos objetos de la misma clase, pero que no están centrados de la misma manera dentro de la imagen como se ilustra enla figura

3.4. A la hora de predecir, la red tendrá un mal desempeño en el intento de comparar las características de la imagen con las que ha aprendido, puesto que los píxeles presentarán una distribución distinta a la que la red aprendió, ocasionando problemas denominados de isotropía.

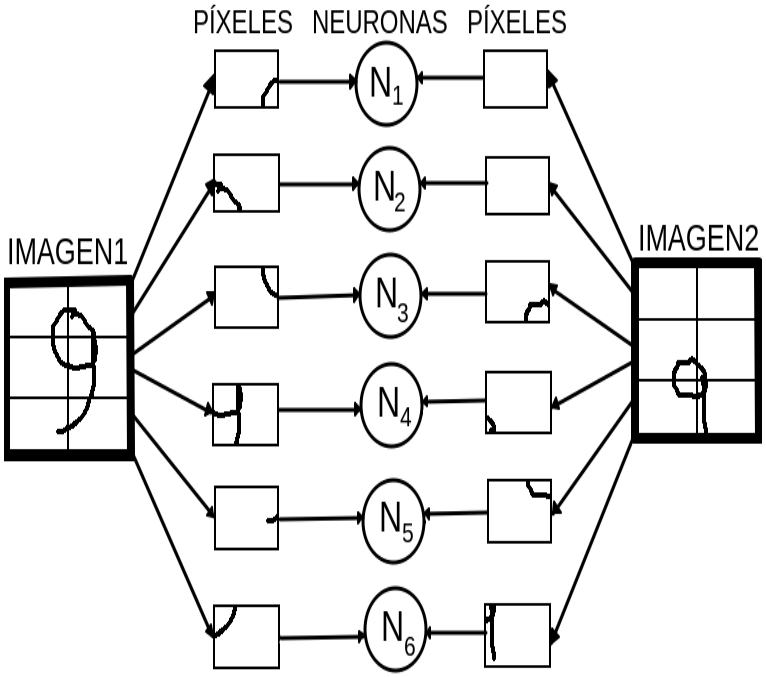


Figura 3.4: Red neuronal clásica comparando imágenes.

Imagen que representa la imposibilidad de una red clásica para entender una imagen de una misma clase debido a la variación en la posición del objeto.

Este problema se resuelve mediante la implementación de la convolución en redes neuronales, ya que brinda la capacidad de extraer características de una forma parecida a como lo hacen los humanos. Es decir, en el ejemplo dado, lo que caracteriza al número 9 no es la información, relación ni la posición que se tiene en términos de píxeles, sino características morfológicas claras que lo distinguen de otros números. Se conforma por un círculo y una línea que se unen de una forma particular; esta característica es inherente al número e independiente del tamaño o la posición de este con respecto a la imagen. La convolución permite a una red neuronal aprender estas características y compararlas de una forma isotrópica.

Supongamos que se quiere distinguir entre un número 9 y un número 8. Para un humano, dada su experiencia, es evidente notar que, mientras el 9 se caracteriza por un círculo y una línea unidos de una forma particular, el número 8 está caracterizado por dos círculos que se unen de una manera específica. Así, para clasificar si la imagen de un número dado es un 8 o un 9, basta con inspeccionar si cumple las características de un 8 o las de un 9 para asignar una probabilidad a cada posibilidad.

Las redes neuronales convolucionales se caracterizan por dos tipos especiales de capas ocultas: la capa de **convolución** y la capa de **agrupación**. Por lo tanto, se entienden como una forma de red neuronal clásica que cuenta con capas especiales que le permiten al modelo aprender características morfológicas de la imagen.

Las funciones de estas capas se definieron tomando como inspiración el comportamiento de las neuronas en la corteza cerebral visual. Estas neuronas permiten al

cerebro humano extraer características de las imágenes percibidas por los ojos, a través de señales eléctricas que llegan a estas neuronas mediante el nervio óptico, el cual se genera a partir de los estímulos oculares. Los científicos descubrieron que la corteza cerebral encargada del procesamiento visual de imágenes se compone de dos tipos distintos de neuronas: las **simples** y las **complejas**.

Las neuronas **simples** se encargan de procesar las señales eléctricas provenientes de los estímulos de una pequeña región del campo visual. Hay muchas neuronas de este tipo que permiten cubrir todo el campo visual. Dada la limitada región del campo visual a la que tienen acceso, estas se activan mediante impulsos eléctricos que se disparan cuando hay presencia de ejes o líneas orientadas de una forma específica.

Las neuronas **complejas** procesan grupos de señales que provienen de varias neuronas simples, por lo que tienen acceso a más información proveniente del campo visual. Estas neuronas se activan mediante impulsos eléctricos cuando provienen de estímulos visuales más complejos, como la orientación de líneas en movimiento en direcciones específicas, independientemente de su ubicación exacta dentro del campo visual.

Dentro de la corteza cerebral encargada del procesamiento visual, existen capas intercaladas de estos dos tipos de neuronas, formando complejas redes neuronales biológicas que brindan al humano la capacidad de análisis visual con la que cuenta, como se ilustra en la figura 3.5.

Para información detallada sobre estas neuronas, véase [8].

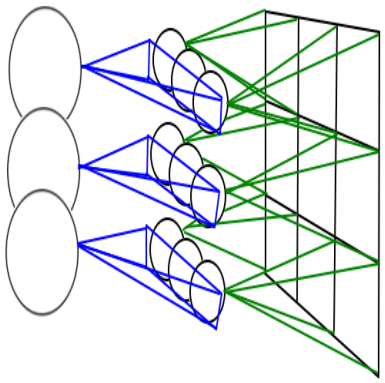


Figura 3.5: Neuronas simples y complejas.

En la imagen, se muestra una capa de neuronas simples, representadas por círculos más pequeños, seguida de una capa de neuronas complejas, representadas por círculos más grandes. El campo de visión completo se subdivide en regiones cuadriculares que representan el acceso a los estímulos para las neuronas simples. A su vez, se muestran los estímulos de múltiples neuronas simples a los que accede una neurona compleja.

¿Cómo se puede emular el funcionamiento de estas neuronas en una red neuronal artificial?

Para lograrlo se requiere del concepto de convolución. Para detectar ejes al igual que las células simples, es necesario analizar una subregión de píxeles de la imagen para determinar si hay un cambio drástico en el valor de los píxeles en esta región

y, así, saber si hay un eje presente. Con esta idea, podemos tomar subregiones de la imagen a través de una submatriz que se denomina núcleo, filtro o ***kernel***. Cada casilla de esta submatriz tiene asignado un número fijo, estos valores numéricos determinan el tipo de filtro que describe esta submatriz.

El proceso de convolución consiste en traslapar la submatriz sobre la imagen y multiplicar cada valor numérico del filtro por el valor del píxel sobre el que está posado. Luego, se procede a sumar el resultado de todas las multiplicaciones resultantes entre el filtro y los valores de píxeles en la imagen para obtener un nuevo resultado que se coloca en una nueva matriz como se ilustra en la figura 3.6.

Este proceso se repite iterativamente moviendo el filtro sobre la imagen un determinado número de píxeles, lo cual se denomina zancada o ***stride***, que se ejecuta de izquierda a derecha y de arriba hacia abajo. De esta forma, según el tamaño y el tipo de filtro que se use, se determina una nueva imagen. Esta nueva imagen presenta transformaciones que permiten detectar ejes de diversas maneras.

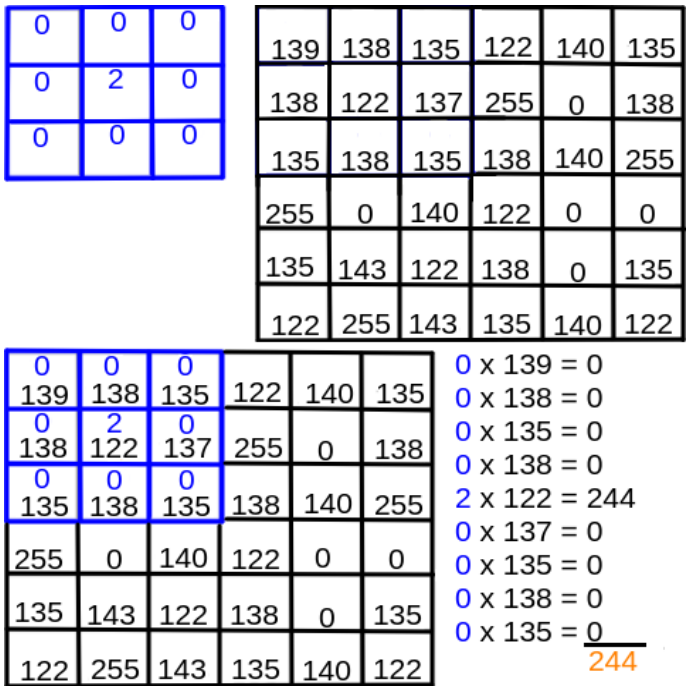


Figura 3.6: Convolución como neurona simple.

La imagen muestra una primera iteración del proceso de convolución que emula una neurona simple. Se utiliza un filtro de 3×3, resaltado en azul, que se superpone a una región de la imagen original.

Para emular el funcionamiento de una neurona **compleja**, utilizaremos otro tipo de operación que nos ayuda a agrupar la información procesada por la capa de convolución y a depender menos de la posición y tamaño de los objetos presentes en la imagen. Esta es la capa de **agrupación**, la cual tiene dos objetivos: reducir la imagen y resaltar las características de interés.

Existen distintas operaciones que se pueden ejecutar en la capa de agrupación, pero una de las más usadas es el de **agrupación máxima**. En este método, a la matriz resultante de la capa de convolución se le sobrepone una matriz de una dimensión menor seleccionada (***kernel***), y de los píxeles de la imagen sobre los que

se superpone, se selecciona el valor más grande. El resto se descarta y se asigna a una nueva matriz, de igual manera que en la convolución; este proceso se ilustra en la figura 3.8.

Se mueve la submatriz a través del parámetro de zancada o *stride*, y el proceso se lleva a cabo en toda la matriz. De esta forma, obtenemos una matriz reducida que codifica las características relevantes de acuerdo al filtro usado y descarta el resto.

El **relleno** o ***padding*** es una técnica que permite la aplicación de filtros a matrices, permitiendo modificar las dimensiones de estas mismas para poder hacer cálculos matriciales consistentes en los bordes. Se logra agregando marcos de valores constantes a las dimensiones de las matrices, como se ilustra en la figura 3.7.

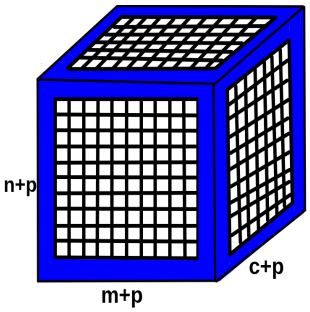


Figura 3.7: Relleno.
Imagen que ilustra el relleno de p píxeles en cada dimensión de una matriz.

Al igual que en las redes neuronales biológicas, las redes neuronales convolucionales permiten extraer características de interés en las imágenes a través de una iteración de capas convolucionales, de agrupación y de capas de procesamiento, como en las redes neuronales clásicas. Dicho esto, a pesar de que hay una gran variedad de filtros para elegir con respecto a la convolución y la agrupación, no hay por qué preocuparse por cuáles elegir, ya que las redes neuronales convolucionales durante el entrenamiento tienen la capacidad de aprender los filtros que mejor se adapten a las características de interés que tenemos sobre los datos.

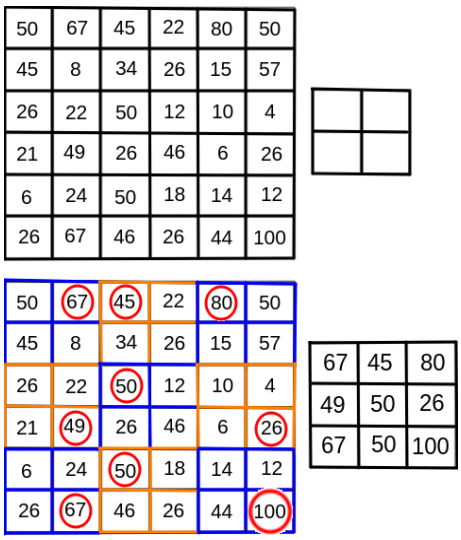


Figura 3.8: Agrupación como neurona compleja.

La imagen muestra una matriz resultante de una convolución a la que se le aplica una agrupación máxima a través de un filtro de 2×2. La alternancia de colores muestra el recorrido del filtro, resaltando con un círculo rojo el valor máximo que conserva la operación. Finalmente, se muestra la matriz reducida resultante que contiene estos valores máximos.

Formalmente, la convolución es un operador en el espacio de funciones continuas $* : C(\mathbb{R}) \times C(\mathbb{R}) \to C(\mathbb{R})$ que asigna dos funciones a una tercera que representa la magnitud en la que se superponen y una versión trasladada e invertida de la segunda mediante

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t - \tau)d\tau$$

Esta definición puede adecuarse a espacios discretos mediante

$$f[m] * g[m] = \sum_{n} f[n]g[m - n]$$

De esta manera, la convolución puede ser extendida a matrices. Así, dadas dos matrices $A, B \in M_n(\mathbb{R})$ la convolución $A * B$ esta definida por

$$(A * B)_{ij} = \sum_{k=1}^{p} \sum_{l=1}^{q} A_{i-k+r,j-l+s} \cdot B_{kl}$$

De forma general, dada una matriz $A_{n \times m \times c}$, al aplicarle algún relleno de ancho p, resulta en la matriz $A_{n+p \times m+p \times c+p}$ y, dado un filtro $F_{t \times t \times t}$ con salto s, la convolución produce una matriz de dimensiones definidas por

$$A_{n+p \times m+p \times c+p} * F_{t \times t \times t} = B_{\lfloor \frac{n+p-f}{s}+1 \rfloor \times \lfloor \frac{m+p-f}{s}+1 \rfloor \times \lfloor \frac{c+p-f}{s}+1 \rfloor}$$

Donde $\lfloor \cdot \rfloor : \mathbb{R} \to \mathbb{Z}$ es la función **piso**, definida por $\lfloor x \rfloor = \text{máx}\{m \in \mathbb{Z} \mid m \leq x\}$.

Para más información sobre redes convolucionales, véase [9].

3.3. Arquitecturas para la detección en imágenes

En esta sección se da una descripción de las arquitecturas utilizadas en el desarrollo de la presente investigación.

3.3.1. YOLO

Y.O.L.O: *You Only Look Once*, por su nombre en inglés que se traduce como solo necesitas mirar una vez al español, es una familia de modelos de visión artificial basados en convoluciones que se utilizan en diversas tareas, como la detección, segmentación y seguimiento de objetos, entre otros.

Existen implementaciones de estos modelos en cuatro tamaños de acuerdo a su profundidad: pequeña (s), mediana (m), grande (l) y extra grande (x), cada una de las cuales proporciona tasas de precisión mAP progresivamente más altas. Cada variante también requiere un tiempo diferente para ser entrenada.

Arquitectura de Y.O.L.O

Esta arquitectura hace uso de la generación de características a partir de imágenes de entrada. Posteriormente, estas características se introducen en un sistema de predicción que crea cuadros alrededor de los objetos y anticipse compone de tres partes principales.

Columna vertebral: Se trata de una red neuronal convolucional que extrae características de la imagen a diferentes niveles de granularidad, es decir, a distintos niveles de detalle.

Cuello: Son una serie de capas que pueden mezclar y combinar las características extraídas de las imágenes para pasarlas a una predicción.

Cabeza: Accede a las características del cuello y lleva a cabo la regresión, lo cual conduce a la predicción de los cuadros delimitadores y la clase del objeto presente en la imagen.

YOLOv8 incluye técnicas de aumento de datos que aplica transformaciones geométricas a los datos para una mayor variedad, su columna vertebral esta definida por una red DenseNet [28].

Detección de objetos de Y.O.L.O

Para llevar a cabo la detección de objetos, el algoritmo comienza dividiendo la imagen en una cuadrícula de $n \times n$ subregiones.

Para cada subdivisión, se generan N posibles cuadros delimitadores con el objetivo de detectar objetos, estos cuadros se ponderan según la probabilidad de encerrar un objeto. De esta manera, se calculan un total de $n \times n \times N$ cuadros delimitadores, los cuales son posteriormente discriminados mediante un umbral mínimo de probabilidad.

Los cuadros que no fueron descartados por el umbral pasan por un segundo proceso denominado supresión de no máximos, en el cual, a partir de los cuadros delimitadores con la ponderación más alta, se realiza la unión que elimina los traslapes existentes entre ellos para mejorar la detección, como se ilustra en la figura 3.9.

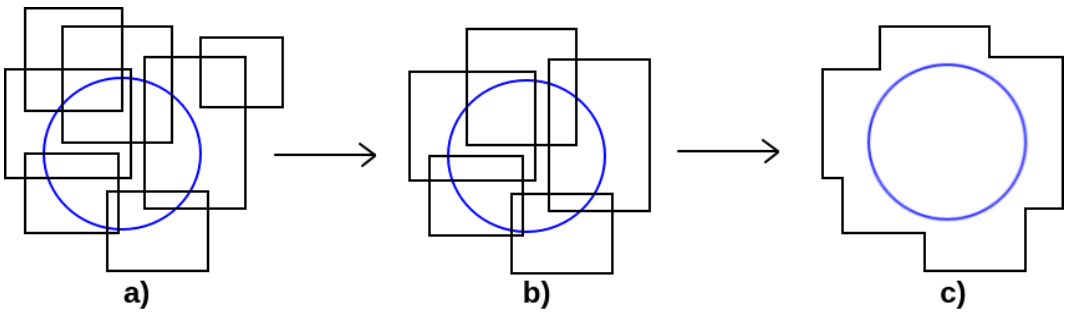


Figura 3.9: Supresión de no máximos.

En la magen se ilustra el proceso de supresión de no máximos para la detección del objeto representado por el círculo azul, utilizada por Y.O.L.O. para mejorar la detección de objetos. a) Muestra los recuadros delimitadores inicialmente propuestos por el modelo. b) Muestra los recuadros delimitadores después de eliminar los

menos probables. c) Muestra la imagen de la unión de los traslapes de los recuadros más probables.
Finalmente, se obtienen los cuadros delimitadores con una probabilidad más alta de contener un objeto, que posteriormente pasan a ser clasificados.

Para obtener un artículo e información completa sobre Y.O.L.O., véase [12], [33] y [13].

3.3.2. RetinaNet

Es una arquitectura de red neuronal profunda diseñada para la detección de objetos en tiempo real. Con una columna vertebral basada en ResNet [29] y una implementación de anclajes (*anchors*) que permiten predecir la clase y la ubicación de los objetos presentes en una imagen, esta arquitectura se compone de cuatro módulos principales que le otorgan su funcionalidad.

- **Columna vertebral base:** Se utiliza ResNet para generar un mapa de características de la imagen analizada.
- **Generador de anclas:** Este funciona para crear cuadrículas de tamaño y escala predefinidos sobre la imagen original, las cuales se denominan anclas. Estas anclas son puntos de referencia utilizados en la predicción de la ubicación de los objetos presentes en la imagen.
- **Red clasificadora:** Es la parte de la arquitectura que permite predecir la probabilidad de que cada ancla se asocie a una clase específica a través de una función softmax.
- **Red de regresión:** Es la parte de la arquitectura encargada de predecir el tamaño y la ubicación de cada objeto asociado a un ancla, lo anterior a través de una regresión lineal que proporciona las coordenadas.

Para información detallada sobre RetinaNet, véase [66].

Capítulo 4

Aprendizaje profundo para imágenes de COVID19

En este capítulo se presentan investigaciones que documentan resultados y esfuerzos para resolver tareas relacionadas con los objetivos de la presente investigación. En general, se abordan estudios que se centran en la detección y clasificación de COVID19 mediante el uso de imágenes médicas y técnicas de aprendizaje profundo.

4.1. Aprendizaje profundo para clasificación COVID19

En la investigación reportada en el artículo titulado *Detection and analysis of COVID19 in medical images using deep learning techniques* [19] por Dandi Yang, et al., se utilizaron cuatro modelos de redes neuronales convolucionales preentrenadas para aplicar transferencia de conocimiento, con el fin de mejorar la clasificación en tomografías computarizadas y radiografías para la detección de COVID19. Los modelos comparados son:

- VGG16
- DenseNet121
- ResNet50
- ResNet152

Estos modelos se entrenaron en tres conjuntos de datos distintos:

- *COVID-19 Radiography Database*, proporcionado por *Kaggle*, que contiene imágenes de casos positivos de COVID-19 junto con imágenes de neumonía normal y viral.
- *Labeled Optical Coherence Tomography and Chest X-Ray Images for Classification*, que contiene radiografías de tórax clasificadas como normales y con algún tipo de neumonía.
- *SARS-CoV-2 CT Scan Dataset*, que contiene 1252 tomografías computarizadas diagnosticadas como positivas para la infección por SARS-CoV-2 y 1229 tomografías computarizadas de pacientes sanos.

Entre los resultados más destacados, se reporta que la exactitud y la puntuación F1 de ResNet fueron del 96 % en el diagnóstico de COVID-19 para tomografías computarizadas, y se alcanzó una exactitud del 99 % con VGG16 en la detección de imágenes de rayos X de COVID-19 y neumonía.

Por otro lado, los autores muestran que aplicar aprendizaje por transferencia ayuda a superar problemas de sobreajuste debido a la cantidad limitada de imágenes de entrenamiento en el aprendizaje profundo.

En la investigación reportada en el artículo *An efficient mixture of deep and machine learning models for COVID19 diagnosis in chest X-ray images* [16] por Wang D. et al., se presenta un método para el diagnóstico automático de COVID19 utilizando imágenes de rayos X de tórax.

Se emplean cinco redes neuronales preentrenadas: VGG16, InceptionV3, ResNet50, DenseNet121 y Xception, junto con un algoritmo de selección de características para extraer las características significativas de las imágenes de pacientes con COVID19. Además, se prueban distintos modelos clásicos de aprendizaje automático como clasificadores.

Los autores proponen una metodología que se divide en tres etapas.

- **Preprocesamiento:** Las imágenes se redimensionan a 224×224 píxeles, se normaliza la intensidad de los píxeles en un rango de $[-1, 1]$, y se aplica aumento de datos mediante rotaciones y *zoom* aleatorios de 30° y 20 % respectivamente.
- **Aplicación de modelos:** Se utilizan los cinco modelos convolucionales preentrenados en la clasificación de imágenes de objetos cotidianos para extraer características de las imágenes de pacientes con COVID19.
- **Clasificación:** Posteriormente, estas imágenes son evaluadas por distintos clasificadores de aprendizaje automático (árboles de decisión, bosque aleatorio, *Adaptive Boosting*, *bagging* y máquinas de vectores de soporte). Finalmente, se evalúan las combinaciones de estos métodos para realizar una comparativa.

Para estas tareas, se utilizan tres conjuntos de datos distintos.

- *COVID19 Chest X-ray Dataset Initiative*, que incluye más de 657 imágenes de rayos X y tomografías computarizadas de pacientes infectados con COVID19 y otras enfermedades.
- *COVID19 Image Data Collection*, de donde se seleccionaron 37 imágenes de rayos X de pacientes con COVID19.
- *Chest X-ray*, de donde se utilizaron aleatoriamente 565 imágenes de rayos X normales del conjunto de datos de imágenes de rayos X de tórax proporcionado por *Kaggle*.

Los autores reportan que el modelo Xception muestra una exactitud del 96.75 % y un puntaje F1 promedio de 96.38 %, con una alta especificidad del 99.17 % y una sensibilidad del 94.16 %. En contraste, se encontró que la mejor combinación del vector de características extraídas y el algoritmo de aprendizaje automático

fue Xception + SVM, logrando una exactitud del 99.33 %. En general, el método propuesto alcanzó una exactitud del 95 %.

En la investigación se enfatiza que la razón del buen desempeño del modelo es que Xception utiliza convoluciones separables en profundidad para reemplazar la operación de convolución original en InceptionV3.

Finalmente, en la investigación reportada en el artículo *IKONOS: an intelligent tool to support diagnosis of COVID19 by texture analysis of X-ray images* [15], por Juliana C. Gomes et al., se propone un sistema inteligente de apoyo al diagnóstico por imágenes de rayos X.

Se propone el desarrollo de IKONOS, una aplicación de escritorio para apoyar y optimizar el diagnóstico de COVID19 a través de imágenes de rayos X de tórax, con una herramienta de fácil mantenimiento y escalabilidad, utilizando algoritmos de baja complejidad computacional para la minimización de costos.

El conjunto de datos consiste en 170 imágenes de rayos X de tórax de pacientes con COVID19, neumonía viral y neumonía bacteriana.

Se extraen características de textura de las imágenes utilizando los momentos de Haralick [23] y Zernike [24]. Posteriormente, se entrenaron y evaluaron diferentes clasificadores que incluyen K-vecinos cercanos, máquina de soporte vectorial, bosques aleatorios y redes neuronales.

Esta investigación permite concluir que, aun con una pequeña cantidad de datos, las redes neuronales profundas tienen la capacidad de diagnosticar COVID19, ya que muestran una precisión del 94 %, que es superior a la de los otros modelos analizados, para los cuales la precisión está en el rango del 88 % al 92 %. Esto nos da un referente de la viabilidad del uso de redes neuronales para la identificación y detección de COVID19.

4.2. Diagnóstico y localización de COVID19

Hasta ahora se han presentado algunas investigaciones sobre el uso de aprendizaje profundo para la clasificación en el diagnóstico de COVID19. En esta sección, se presentan investigaciones sobre modelos de aprendizaje profundo que no solo diagnostican la enfermedad, sino que también localizan las regiones afectadas en las imágenes médicas, lo cual es uno de los objetivos del presente trabajo.

En la investigación documentada en el artículo *COVID19 lesion discrimination and localization network based on multi-receptive field attention module on CT images* [7] por Xia Ma et al., se propone una red con un módulo de atención de campo multirreceptivo para diagnosticar COVID19 en imágenes de tomografía computarizada. Este módulo de atención incluye tres partes.

- PCM: Módulo de convolución piramidal que puede mejorar la capacidad de diagnóstico de la red para lesiones de diferentes tamaños y formas.
- SAB: Bloque de atención espacial de campo multirreceptivo.

- CAB: Bloque de atención de canal de campo multirreceptivo que, en conjunto con SAB, permite enfocar las características extraídas de la red en el área de la lesión para mejorar la capacidad de discriminación y localización de COVID19.

El modelo propuesto utiliza VGG16 como base. A diferencia de VGG16, se reemplaza la capa de convolución antes de cada capa de agrupación con cada módulo de atención de campo multirreceptivo propuesto y se evalúa su desempeño.

El modelo se entrenó en un conjunto de datos compuesto por 200 imágenes de tomografía computarizada (TC) de tórax, de las cuales 100 correspondían a pacientes con COVID19 y 100 a pacientes sin COVID19.

Los resultados mostraron una mejora acorde con la implementación de cada módulo propuesto, ya que se reportó que la red VGG16 por sí sola alcanzó una exactitud del 93.02 %, la cual ascendió de manera monótona a 97.12 % con la implementación de cada módulo, reflejando la efectividad de estos.

Finalmente, la investigación documentada en el artículo *A deep learning approach for COVID19 screening and localization on chest x-ray images* [5], por Karem Daiane Marcomini et al., es de especial relevancia, ya que utiliza la base de datos pública *SIIM-FISABIO-RSNA* [20], la cual contiene imágenes de rayos X anotadas con daño pulmonar causado por COVID19, y esta base de datos juega un papel fundamental en el presente trabajo.

En esta investigación se propone un modelo de aprendizaje que puede discernir entre dos clases de las cuatro propuestas en la base de datos, así como las áreas de opacidad pulmonar referentes a estas clases. Dado que las clases están desequilibradas (el 69,2 % de las imágenes son COVID19 positivas), se aplicó un aumento de datos a las imágenes de la categoría sanos.

El conjunto de datos se dividió en conjuntos de entrenamiento y prueba con una proporción de 90:10, y para la clasificación se aplicó una validación cruzada de 5 veces al conjunto de entrenamiento. Para realizar la clasificación, se utilizó la arquitectura **EfficientNetB4** [4], mientras que YOLOv5 [13] se empleó para la tarea de detección.

Los datos se reorganizaron en dos categorías de imágenes del conjunto de datos: apariencia positiva para COVID19 y negativa para neumonía. Se agruparon las apariencias típicas e indeterminadas en la clase positiva de COVID19, y se eliminaron las imágenes relacionadas con otros tipos de neumonía (etiquetadas como atípicas), ya que es probable que las apariencias típicas e indeterminadas correspondan a una infección por COVID19.

Al clasificador basado en la arquitectura EfficientNetB4 se le reemplazó por una operación de agrupación promedio global, seguida de una normalización por lotes y una capa densa con una neurona que utiliza la función de activación sigmoide.

El entrenamiento se realizó con 30 épocas en lotes de 16 imágenes por paso, utilizando un optimizador Adam con una tasa de aprendizaje de 0.0001. Se utilizó el aumento de datos durante el entrenamiento y todas las imágenes se redimensionaron a 380x380 píxeles y se aplicó un procesamiento en el que se normaliza el

histograma de intensidades.

Para la localización se utilizó la red YOLOv5 con una resolución de entrada de 512×512. Se entrenó durante 50 épocas con un tamaño de lote de 8.

Posterior a la salida de los detectores de objetos, se eliminaron los cuadros delimitadores superpuestos obtenidos durante la etapa de predicción mediante el algoritmo de supresión no máxima (NMS), utilizando un valor umbral de 0.5.

Las métricas para el desempeño del detector de opacidades se resumen en la tabla 4.1.

Conjunto	**VP**	**FP**	**FN**	**mAP**
Diagnosticado COVID19	495	169	254	59.51
Predicho COVID19	478	226	271	53.57

Tabla 4.1: Evaluación del detector.

Tabla recreada de [5], donde se muestran las métricas de evaluación para el modelo de detección.

El modelo utilizado para la clasificación presentó métricas que reflejan un desempeño satisfactorio en la tarea para la que fue entrenado, las cuales se resumen en la tabla 4.2.

Fold	**VP**	**FN**	**VN**	**FP**	**Exactitud**	**Precisión**	**Recall**	**F1**	**AUROC**
0	373	24	92	70	0.832	0.842	0.940	0.888	0.883
1	348	49	105	57	0.810	0.859	0.877	0.868	0.856
2	364	33	95	67	0.821	0.845	0.917	0.879	0.862
3	367	30	103	59	0.841	0.862	0.924	0.892	0.893
4	360	37	106	56	0.834	0.865	0.907	0.886	0.887
μ	362.4	34.6	100.2	61.8	0.828	0.855	0.913	0.883	0.876
σ	8.4	8.4	5.6	5.6	0.011	0.009	0.012	0.008	0.015

Tabla 4.2: Evaluación de la clasificación.

Tabla recreada de [5], donde se condensan las métricas relacionadas con la clasificación dada por el modelo.

Esto marca un antecedente directo en la búsqueda de soluciones para el problema propuesto en la presente investigación y nos proporciona una visión general de los desafíos que representa.

4.3. Comentarios sobre las investigaciones antecedentes

Se han expuesto diversas investigaciones centradas en el desarrollo de tecnologías basadas en el análisis automático de imágenes para la ejecución de tareas relacionadas con el COVID19 a través de imágenes médicas, como radiografías y tomografías computarizadas. En estas investigaciones, se destacan distintos puntos que hacen de la presente investigación un esfuerzo plausible.

Por una parte, se plasmó la eficacia de la implementación de modelos en el diagnóstico de la enfermedad mediante la clasificación de imágenes de pacientes sanos y enfermos, lo cual nos brindó un antecedente positivo en el uso de imágenes médicas para el desarrollo de herramientas tecnológicas que buscan automatizar procesos relacionados con el COVID19.

Por otro lado, se mostraron diversos esfuerzos por mejorar el desempeño de los modelos a través de la modificación de arquitecturas para la implementación de módulos de atención, así como de capas de convoluciones separables en profundidad. Esto proporcionó un panorama general de la constante búsqueda de mejoras en el rendimiento de los modelos, que a menudo están limitados por las condiciones de los datos de entrenamiento. Esta situación motivó la búsqueda de implementaciones que contribuyan al desempeño de tareas aprendidas por modelos de aprendizaje profundo.

Finalmente, se presentó un antecedente directo que colapsa las clases del conjunto de entrenamiento en dos categorías: sanos y neumonía por COVID19. Este enfoque aborda el problema de localización e identificación de daños, lo cual representa una tarea parcial en comparación con la explorada en el presente trabajo, ya que aquí se buscó discernir entre las cuatro posibles clases de lesiones. No obstante, este antecedente ofrece un panorama amplio de los retos y necesidades que rodean el problema abordado, al mismo tiempo que sustenta la necesidad de buscar mejoras para el desempeño de los modelos empleados en esta tarea.

Así concluye este capítulo, en el que se expusieron distintas investigaciones que, aunque se centran en diversas tareas relacionadas con la enfermedad COVID19 y la aplicación de aprendizaje profundo, hasta el momento del desarrollo del presente trabajo, solo se cuenta con una investigación que aborda parcialmente la tarea en la que nos hemos enfocado: la localización e identificación de daños pulmonares causados por COVID19.

Además, los aportes incentivados por la competencia *Kaggle*, que proporcionó el conjunto de datos públicos utilizado para esta investigación, resaltan la posibilidad de explorar nuevos mecanismos y propuestas para mejorar el desempeño en dichas tareas.

Capítulo 5

Métodos y materiales

En este capítulo se presentan las acciones y propuestas planteadas para alcanzar el objetivo principal de la presente investigación; de igual forma, se detallan los materiales esenciales empleados en el desarrollo de la misma.

5.1. Recursos computacionales

En esta sección, se proporcionan todos los detalles sobre la computadora utilizada para el almacenamiento y exploración de datos, así como para el entrenamiento de modelos.

Para gestionar los grandes volúmenes de imágenes y sus anotaciones, que superan 1 TB (terabyte), equivalente a 1000 GB, así como para su exploración, visualización y preparación, se requiere una computadora con una gran capacidad de almacenamiento interno. Además, dada la necesidad de entrenar modelos de aprendizaje profundo, es indispensable que esta cuente con una alta capacidad de cómputo. Para estos fines, se utilizó una computadora **Dell Inc. Precision Tower 7910**, perteneciente al Laboratorio Nacional de Microscopía Avanzada (IBt, UNAM), modificada con las siguientes especificaciones.

- **Memoria:** 128 Gb.
- **Procesador:** Intel® Xenon® E5-2630 v3 × 32
- **GPU:** NVIDIA GeForce RTX™ 3090 Ti capacidad 24 GB
- **Capacidad en disco:** 3.9 TB
- **Sistema operativo:** Ubuntu 23.10 64-bit
- **Kernel:** Linux 6.5.0-25-generic

5.2. Conjuntos de datos

Para alcanzar los objetivos de la presente investigación, se utilizan tres conjuntos de entrenamiento públicos diferentes, sobre los cuales se realiza un análisis exploratorio con el fin de obtener un conocimiento estadístico detallado de su contenido y estructura, los cuales están disponibles como se describe en el apéndice 7. Los conjuntos de datos utilizados son los siguientes.

RSNA Pneumonia

Proporcionado por Kaggle en el desafío *RSNA Pneumonia Detection Challenge* [1].

Este conjunto contiene 14,863 imágenes de radiografías de tórax de 1024 × 1024 píxeles proporcionadas por tres instituciones distintas.

- *Medical Imaging Data Resource Center (MIDRC)*
- *RSNA International Covid-19 Open Radiology Database (RICORD)*
- Banco de Imagen Médica de la Comunidad Valenciana (BIMCV-COVID-19 Dataset)

Las anotaciones fueron realizadas por un grupo internacional de radiólogos voluntarios, quienes se encargaron de etiquetar la presencia y ubicación de neumonía mediante la delimitación de recuadros en las imágenes.

Para información detallada del conjunto de datos, véase [22].

Chest X-ray14

Contiene imágenes de rayos X de tórax [25], está conformado por 112,120 imágenes de rayos X de tórax de 1024 × 1024 píxeles, de las cuales solo 1000 cuentan con anotaciones de recuadros delimitadores para solo 9 de las 15 clases reportadas en el conjunto.

1. Sin hallazgos: Imágenes sin ningún tipo de los 14 hallazgos médicos.
2. Cardiomegalia: Agrandamiento del corazón.
3. Efusión: Acumulación de líquido entre la pleura y la pared torácica.
4. Neumonía: Condición médica caracterizada por inflamación en los pulmones, donde el oxígeno en los alvéolos es sustituido por líquido.
5. Nódulos pulmonares: Masas sólidas en los pulmones, más pequeñas que masas pulmonares.
6. Atelectasia: Colapso de un pulmón o parte de un pulmón.
7. Neumotórax: Presencia de aire entre los pulmones y la pared torácica.
8. Masa pulmonar: Masa anormal en el pulmón.
9. Infiltración: Inflamación del tejido pulmonar.

Para información detallada del conjunto de datos, véase [25].

SIIM-FISABIO-RSNA COVID19

Este conjunto de datos[2], consta de 6,334 imágenes de radiografías de tórax de diferentes tamaños con anotaciones sobre daños causados por COVID19, proporcionadas por tres instituciones.

[1] Disponible a través de *https://www.kaggle.com/c/rsna-pneumonia-detection-challenge/data*
[2] Disponible a través de *https://www.kaggle.com/c/siim-covid19-detection/data*

- *Medical Imaging Data Resource Center (MIDRC)*
- *RSNA International Covid-19 Open Radiology Database (RICORD)*
- Banco de Imagen Médica de la Comunidad Valenciana (BIMCV-COVID-19 Dataset)

Los datos fueron recopilados por el sistema sanitario de Valencia, España, que se divide en departamentos de salud ubicados en diferentes provincias, las cuales se pueden consultar a través de un mapa interactivo de las regiones donde se tomaron las radiografías[3] [20].

El conjunto de datos se creó utilizando dos bases de datos públicas, **MIDRC-RICORD** [26] y **BIMCV** [20]. Estas proporcionaron radiografías de pacientes positivos y negativos para COVID19. Luego, se eliminaron las imágenes que no tenían etiquetas DICOM asociadas, y se seleccionaron solo imágenes frontales. Además, se llevó a cabo un proceso de anonimización para proteger los datos personales de los pacientes.

El proceso de anotación fue realizado por 22 radiólogos provenientes de América del Norte, América del Sur y Europa. De estos, 9 tenían especialización en tórax y 13 no. Además, 19 de ellos estaban ejerciendo su profesión, mientras que 3 eran residentes de nivel *senior* en radiología.

Los anotadores tuvieron acceso a MD.ai, una herramienta de software de anotación proporcionada por el Centro AIMI para investigadores de Stanford, que ayuda a seleccionar y crear conjuntos de datos de imágenes médicas[4]. Contaron con acceso a capacitaciones para el uso del software mediante conferencias web, así como materiales de referencia y casos de ejemplo para cada categoría anotada.

Veinticinco ejemplos fueron anotados por un radiólogo experto con 15 años de experiencia en radiología y 10 en radiología de tórax. Estas anotaciones se tomaron como referencia para evaluar el desempeño de cada anotador, requiriendo que estos alcanzaran un porcentaje de coincidencia del 60 % o más con el experto. Los anotadores clasificaron las imágenes de acuerdo a los estudios en 4 categorías, siguiendo los estándares registrados en la tabla 5.1.

Se anotaron recuadros delimitadores en las opacidades del espacio aéreo pulmonar para los exámenes clasificados. Se consideraron derrames pleurales, masas/nódulos o neumotórax. Para los casos negativos de neumonía, no se colocaron cuadros delimitadores. Para las opacidades que estaban muy cercanas o adyacentes, los anotadores colocaron un solo cuadro que cubría el área, con el objetivo de reducir la variabilidad, como se ilustra en la figura 5.1.

[3]El cual está disponible a través de https://maigva.github.io/maps/HealthDepartCOVID19.html
[4]Se puede consultar mediante su página https://www.md.ai/

Clasificación de la radiografía	Hallazgos presentes
Apariencia típica	Opacidades bilaterales que muestran fibrosis o volumen pulmonar reducido, centrales y periféricas difusas.
Apariencia indeterminada	Opacidades en la zona superior del pulmón, unilaterales, multifocales y centrales con preservación periférica relativa.
Apariencia atípica	Neumotórax sin características de neumonía, masas o nódulos, neumonía lobar, cicatrización o fibrosis.
Negativo para neumonía	Sin opacidades pulmonares.

Tabla 5.1: Criterios de anotación.

Tabla recreada de [21], donde se muestra el esquema y criterios utilizados para la anotación de datos.

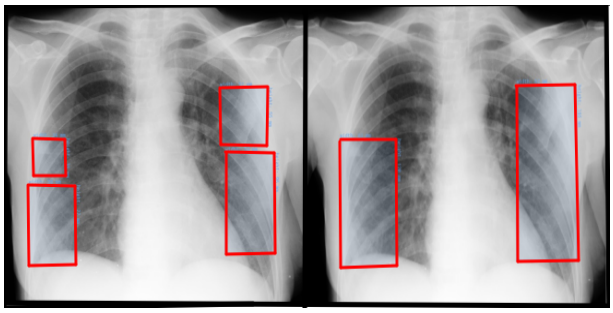


Figura 5.1: Etiquetado de imágenes.

Imagen tomada de [21], donde a la izquierda se muestra una imagen con anotaciones de opacidades cercanas, y a la derecha se presenta la anotación resultante del criterio para el conjunto de datos.

Por último, los autores proponen las clasificaciones de severidad en enfermedades pulmonares, de acuerdo con el número de lesiones encontradas, como se resume en la tabla 5.2.

Clasificación de severidad	Criterio de evaluación
Leve	Opacidades: 1-2 zonas
Moderado	Opacidades: 3-4 zonas
Severo	Opacidades: +4 zonas

Tabla 5.2: Severidades.

Tabla recreada de [27], donde se muestra una clasificación de severidad.

Para obtener información detallada sobre los datos utilizados, consulte [20].

Para detalles sobre la anotación y curación del conjunto de datos, consulte [21].

5.3. Metodología

En esta sección se define la metodología que plasma las estrategias propuestas y se evalúan en el presente trabajo. De esta manera, se proporciona una visión general de la investigación.

Se hace uso de dos modelos de redes neuronales, la implementación de YOLO [13], en la versión 8[5] [42] para determinar el impacto de las estrategias propuestas, así como la implementación de RetinaNet, como se describe en el artículo original[6] [66] para comparar el desempeño brindado por la implementación de las estrategías propuestas en un modelo totalmente diferente con sus respectivas configuraciones inherentes, como por ejemplo una nueva función de pérdida.

En el caso de RetinaNet, se utiliza una función de pérdida total compuesta por la suma de dos subfunciones de pérdida, una para la identificación y otra para la localización. Para la identificación, se emplea la entropía cruzada focal, que asigna mayor peso a los errores en las clases con menor representación en el conjunto de datos. Para la localización, se utiliza una regresión suavizada mediante la regularización L1. En el caso de YOLO, se utiliza una combinación ponderada de dos funciones de pérdida, para la identificación se utiliza la entropía cruzada y para la localización el error cuadrático medio (MSE), las cuales se combinan en una función de pérdida total dada por

$$\text{Total} = \lambda\text{Entropía cruzada} + (1 - \lambda)\text{MSE}$$

Ambas implementaciones se utilizan pre-entrenadas en el conjunto de datos COCO Objects [3], con la finallidad de evaluar el desempeño obtenido en la detección de lesiones médicas a partir de su desempeño en la detección de objetos comunes.

Para todos los conjuntos y entrenamientos, el conjunto de datos se divide en 70 % para entrenamiento, 20 % para validación y 10 % para prueba. Estas asignaciones se realizan asegurando que cada conjunto generado mantenga una distribución proporcional de las clases presentes en el conjunto de datos completo. Los entrenamientos y las decisiones se evalúan con base en el desempeño reportado por las métricas de entrenamiento y validación. La selección final del modelo se lleva a cabo utilizando exclusivamente el conjunto de prueba, con el objetivo de evaluar su desempeño final. Esto se realiza para evitar sesgos en la toma de decisiones relacionadas con el conjunto de prueba durante los entrenamientos.

Los entrenamientos comienzan con el conjunto de datos *RSNA-PNEUMONIA* y sus imágenes reescaladas a 640×640, ya que es un conjunto que contiene información de calidad y está bien estructurado, como se reporta en la subsección 6.1.1. Se utiliza para realizar las primeras evaluaciones y decisiones sobre los modelos en la tarea de detectar la presencia o ausencia de opacidades causadas por neumonía.

Y.O.L.O cuenta con implementaciones de distintos tamaños acorde a la cantidad de parámetros aprendidos por la red. Debido a que la red más pequeña requiere menos tiempo para el proceso de entrenamiento, se utiliza como modelo inicial.

[5]La cual se puede acceder a través del GitHub https://github.com/ultralytics/ultralytics
[6]La cual se puede acceder a través del GitHub https://github.com/yhenon/pytorch-retinanet

Para el proceso de evaluación, se utiliza la métrica mAP50 para la localización e identificación conjunta de opacidades, y las matrices de confusión para la clasificación general de las imágenes, es decir, sin considerar la localización del hallazgo, ya que cada imagen contiene elementos de una sola clase. Finalmente, se analizan las gráficas de las funciones de pérdida obtenidas durante los entrenamientos.

Dado que los conjuntos de datos de entrenamiento solo contienen anotaciones para las imágenes con lesiones, los modelos, en ocasiones, interpretan las imágenes sin etiquetas (sanas) como imágenes de fondo. Por lo tanto, al realizar predicciones sobre imágenes sin lesiones o hallazgos, el modelo puede clasificarlas como imágenes de fondo o como listas vacías de recuadros delimitadores, con una etiqueta de "sin detecciones", como se muestra en la figura 5.2.

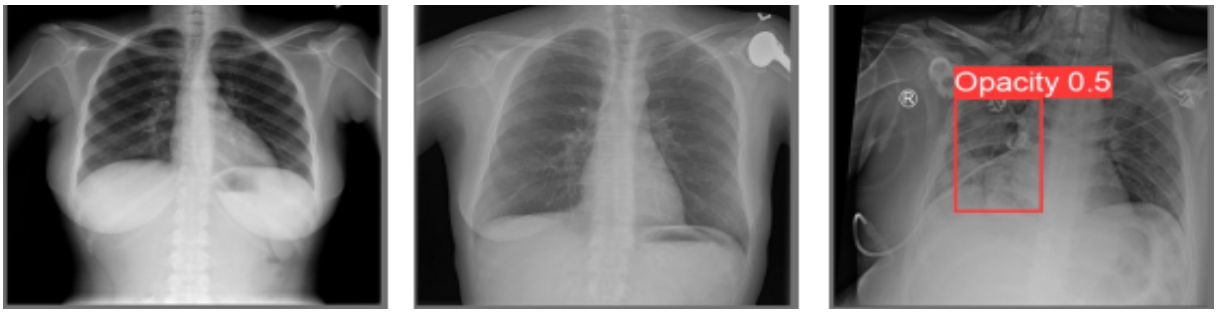


Figura 5.2: Predicciones de imágenes por Y.O.L.O.

Imágenes de prediccion de Y.O.L.O [42]. Las dos primeras imágenes corresponden a pacientes sin opacidades: una presenta la predicción de clase fondo, mientras que la otra no tiene detecciones asignadas. La tercera imagen muestra la presencia de una opacidad con un 0.5 % de confiabilidad.

Para la evaluación de la clasificación y la generación de las matrices de confusión, se mencionan distintos factores que generan diferencias notables entre los elementos de dichas matrices. Las matrices se utilizan para evaluar la clasificación general de las imágenes: si la anotación original contiene un recuadro delimitador de una clase particular y la predicción coincide en la clase del recuadro, se considera como un verdadero positivo. De manera análoga, se definen los demás resultados. Esto contrasta con el mAP, donde además se toma en cuenta la intersección respecto a los recuadros.

Dada la observación anterior sobre las predicciones en imágenes sanas, se consideran ambas predicciones posibles como acertadas para imágenes sin opacidades, es decir, tanto las predicciones sin hallazgos como las listas vacías de recuadros se toman como aciertos para estas imágenes. Este ajuste elimina la clase predefinida *background* de los conteos.

El ajuste en el conteo para la matriz se aplicó en todos los casos donde las imágenes sin objetos de interés no cuentan con ningún recuadro delimitador. Es importante resaltar que este cambio no afecta el rendimiento del modelo, sino únicamente la interpretación asociada a los resultados, como se muestra en la figura 5.3.

Por otro lado, cabe resaltar que la cantidad total de elementos considerados para la generación de las matrices de confusión puede variar entre los entrenamientos de Y.O.L.O y RetinaNet debido a los requerimientos específicos de cada modelo. Mientras que Y.O.L.O permite entrenar el modelo con una lista de recuadros de-

limitadores asignados a una sola imagen, RetinaNet requiere un enfoque distinto, ya que solo admite un recuadro delimitador por imagen. Esto implica que, para imágenes con múltiples recuadros, es necesario incluir la imagen varias veces en el entrenamiento, una por cada recuadro presente.

Esta diferencia genera variaciones en los conteos totales entre ambos modelos. Sin embargo, los resultados son comparables al analizar los porcentajes de aciertos respecto a los totales. Por otro lado, cuando se varían las anotaciones en los datos de entrenamiento, factores similares afectan la cantidad de conteos totales. Por ejemplo, en el caso de imágenes de pacientes sin hallazgos, si se incluye un recuadro para cada pulmón, estas imágenes contarán con dos anotaciones por imagen, lo que afecta los conteos totales de manera similar para los entrenamientos de Y.O.L.O y RetinaNet.

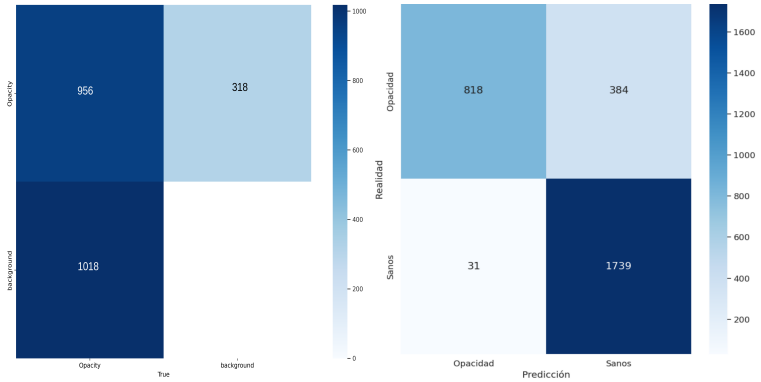


Figura 5.3: Modificación de matrices de confusión.

Imagen que muestra, a la izquierda, la matriz de confusión generada por Y.O.L.O [42] con la clase fondo, la cual genera ruido en la interpretación de los resultados. A la derecha, se presenta la misma matriz de confusión, pero con el conteo propuesto.

La implementación de Y.O.L.O [42] normaliza automáticamente las imágenes durante el proceso de entrenamiento, de acuerdo con lo reportado en el código correspondiente *dataloader.py*. Esto implica que no es necesario normalizar manualmente las imágenes antes de entrenar el modelo.

Por otro lado, la implementación utilizada de RetinaNet [43] también aplica normalización durante el proceso de entrenamiento, como se indica en el código correspondiente *dataloader.py*. Esta normalización sigue el formato estándar utilizado para modelos preentrenados en ImageNet.

5.3.1. Exploración de hiperparámetros

Se comienza por explorar los hiperparámetros que ofrece el modelo[7] para determinar cuáles influyen con mayor impacto en el desempeño.

Se inicia con una elección inicial simple de los hiperparámetros, como se muestra en la tabla 5.3. Esto permite un entrenamiento que sirve como base para medir la implementación de futuras decisiones, y permite explorar cómo se comporta el

[7]Los cuales se pueden consultar a través de https://docs.ultralytics.com/es/modes/train/#augmentation-settings-and-hyperparameters

modelo durante las épocas de entrenamiento. Se emplea la técnica de paciencia, que guarda el modelo en el punto de mejor desempeño. Además, si no se observa una mejora en las métricas después de cierto número de épocas definidas (paciencia), permite detener el entrenamiento en fases de no mejora prolongada.

La elección del tamaño de lote se realiza de manera automática y se determina de acuerdo con la disponibilidad de memoria y capacidad de procesamiento. El tamaño de lote utilizado para cada entrenamiento, así como el tiempo de ejecución requerido, se reportan en las tablas 6.2 y 6.3.

Hiperparámetro	**Valor**
epochs	350
patience	100
optimizer	auto
augment	False
auto_augment	None

Tabla 5.3: Hiperparámetros iniciales propuestos para Y.O.L.O v8.

Dado el entrenamiento anterior como referencia, se procede a explorar el espacio de hiperparámetros, probando las opciones que ofrece el modelo para el aumento de datos con el fin de determinar cuál ofrece los mejores resultados.

- **Aumento automático:** Prueba en un pequeño conjunto de datos qué transformaciones funcionan mejor para los datos de entrenamiento dada una función evaluadora y selecciona las que mejor puntuación obtienen.
- **Aumento aleatorio:** Toma transformaciones aleatorias entre las que se encuentran traslaciones, rotaciones, recortes, etc. Las cuales se aplican al conjunto de entrenamiento.
- **Aumento mixto:** Consiste en tomar composiciones de transformaciones individuales y aplicarlas al conjunto de entrenamiento.
- **Aumento personalizado:** Se hace una elección específica de los hiperparámetros que definen las transformaciones que se aplican sobre el conjunto de entrenamiento.

Para el aumento personalizado, se propone la elección de los hiperparámetros reportados en la tabla 5.7.

Tomando como punto de partida el entrenamiento con mejores resultados, se indaga cuál optimizador es el que brinda mayor impacto. Dentro de las opciones a probar tenemos SGD, RMSProp, Adam, Adamax, RAdam, NAdam y AdamW.

Hasta ahora se utiliza SGD, el cual es seleccionado automáticamente por el modelo. Con respecto a cómo se han desarrollado estos algoritmos, se prueban ordenadamente de acuerdo a sus mejoras:

Comenzando con SGD, que realiza una selección aleatoria de vectores para calcular el gradiente de la función de pérdida y actualiza los parámetros de acuerdo

a una tasa de aprendizaje constante. Luego, RMSProp parte de esta idea para utilizar el promedio del cuadrado de los vectores empleados para penalizar la tasa de aprendizaje y eliminar su naturaleza constante.

Adam da un paso más e implementa el uso del primer momento (magnitudes de los vectores) y el segundo momento (magnitudes elevadas al cuadrado de los vectores) para penalizar la tasa de aprendizaje. Por otro lado, Adamax incluye en el segundo momento la norma del máximo.

RAdam aplica una rectificación a la tasa de aprendizaje con respecto a su valor en la iteración anterior, mientras que NAdam aplica aceleración de Nesterov, que consiste en tomar en cuenta la dirección del segundo momento anterior en el cálculo del actual. Finalmente, AdamW aplica un factor de decaimiento respecto a los parámetros, lo que penaliza el segundo momento.

Posteriormente, se explora el impacto de aplicar distintas técnicas de aprendizaje automático, como el uso de una tasa de aprendizaje cosenoidal, la cual ayuda a disminuir oscilaciones bruscas en la asignación de la tasa de aprendizaje, dando paso a una asignación más suave de valores.

Dropout, que en el proceso de entrenamiento permite desactivar aleatoriamente neuronas de las capas que conforman la red, es útil para evitar sobreajustes e introducir variabilidad en el proceso de entrenamiento. También se modifica el parámetro que da peso a la granularidad de la clasificación del modelo, y finalmente se especifica una clase única, probando así la implementación independiente de cada uno de los hiperparámetros propuestos en la tabla 5.4.

Hiperparámetro	**Valor**
dropout	0.5
cos_lr	True
dfl	2.5
single_cls	True

Tabla 5.4: Hiperparámetros de prueba propuestos para Y.O.L.O v8.

Al comparar los rendimientos ofrecidos por la exploración de los hiperparámetros en cada uno de estos entrenamientos, se procede finalmente a realizar un primer entrenamiento de la arquitectura RetinaNet, siguiendo la idea de una rutina sencilla como se muestra en la tabla 5.5 y utilizando el mismo conjunto de datos. De esta forma, se obtiene una base de comparación para evaluar el impacto de las estrategias exploradas, así como el desempeño entre estas dos arquitecturas.

Hiperparámetro	**Valor**
depth	50
epochs	160
optimizer	SGD

Tabla 5.5: Hiperparámetros iniciales propuestos para RetinaNet.

5.3.2. Cambios para los datos

En este punto, con base en el mejor entrenamiento obtenido, se procede a explorar el impacto que tiene en las métricas el realizar cambios en los datos y anotaciones de entrenamiento.

La exploración de hiperparámetros se realiza con las imágenes redimensionadas a 640 × 640 píxeles. Dada la naturaleza de las características finas respecto a las opacidades, se propone evaluar un entrenamiento con imágenes en su tamaño original de 1024 × 1024 píxeles.

Por otro lado, se explora el entrenamiento con distintas anotaciones para el caso de imágenes sin presencia de opacidades. De esta forma, no solo se influye en las métricas de identificación, sino también en las de localización. Al no contar con anotaciones, el desempeño de esta clase no afecta las métricas, además de permitir evaluar cómo influye delimitar las regiones de interés. Se propone explorar los siguientes etiquetados, además de las anotaciones originales, las cuales equivalen a recuadros delimitadores de ancho y alto cero:

- Para imágenes sin presencia de opacidades, se etiqueta un recuadro delimitador que abarca todo el contorno de la imagen, como se ilustra en la figura 5.4.
- Un segundo etiquetado busca centrar la atención en las regiones de los pulmones, anotando los pulmones sanos, cada uno dentro de un recuadro individual, como se ilustra en la figura 5.5.

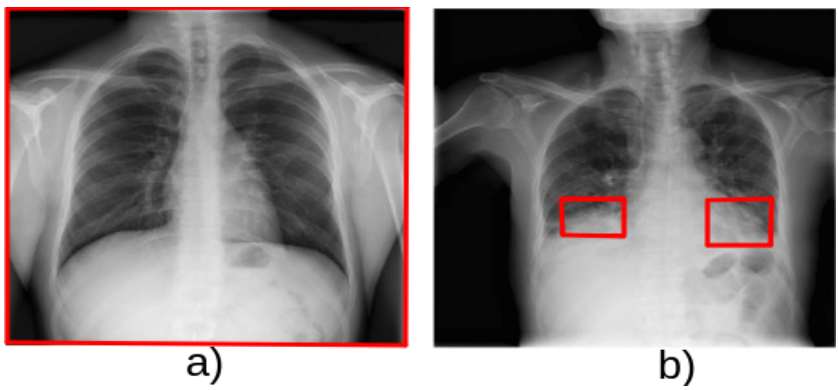


Figura 5.4: Anotaciones propuestas.

Imagen tomada de [21]. a) Muestra una imagen de una persona sana, con la anotación propuesta. b) Imagen que muestra anotaciones referentes a la presencia de opacidades en los pulmones.

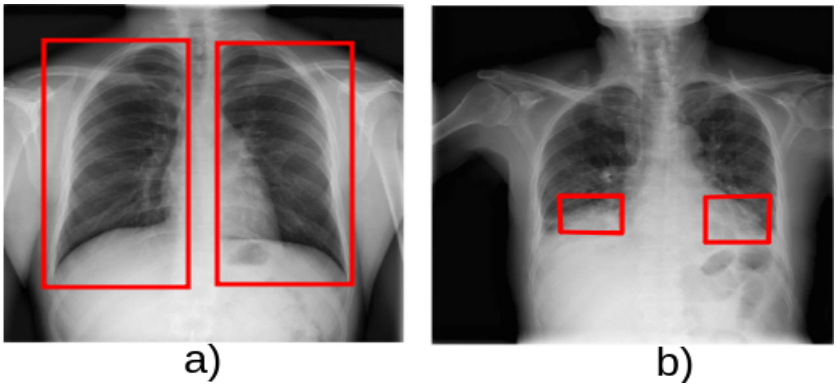


Figura 5.5: Anotaciones propuestas.

Imagen tomada de [21]. a) Muestra una imagen de una persona sana, con la anotación propuesta. b) Imagen que muestra anotaciones referentes a la presencia de opacidades en los pulmones.

5.3.3. Profundidad de la red

En este punto, aplicando las estrategias que reporten un mayor impacto en el desempeño del modelo *nano*, se procede a explorar el impacto que tiene la profundidad del modelo para Y.O.L.O. Así, se propone entrenar cada uno de los tamaños disponibles, los cuales se muestran en la tabla 5.6, para determinar cuál es el de mayor eficacia.

Tamaño	**mAP**	**Parámetros**
Nano	37.3	3.2 millones
Small	44.9	11.2 millones
Medium	50.2	25.9 millones
Large	52.9	43.7 millones
Xtra large	53.9	257.8 millones

Tabla 5.6: Tamaños de Y.O.L.O v8.

Tabla recreada de [42], donde se muestran los tamaños disponibles de la arquitectura, el mAP obtenido en su preentrenamiento y el número de parámetros en la red.

5.3.4. Modificación de la red

Se procede a agregar módulos de atención que, para tomografías computarizadas, mostraron una mejora significativa en el rendimiento del modelo y en la segmentación de regiones pequeñas, tal como se documenta en el artículo *COVID19 lesion discrimination and localization network based on multi-receptive field attention module on CT images* [7], con el fin de medir su impacto en el desempeño.

La implementación de los módulos se puede resumir de la siguiente manera:

Suponiendo que el mapa de características de entrada dado es $F \in \mathbb{R}^{C\times H\times W}$, donde los conjuntos de mapas de características de campo receptivo múltiple $F_N = \{F_j\}_{j=1}^{n}$, tal que $F_j \in \mathbb{R}^{C\times H\times W}$ para cada $1 \leqslant j \leqslant n$, se obtienen al pasarlos por la capa **PCM**.

Luego, F_N se envía a dos bloques de atención por separado. En primer lugar, se suman y se envían al **SAB** para obtener el peso de atención espacial $W_S \in \mathbb{R}^{H \times W}$. En segundo lugar, se concatenan F_N y luego se envían al **CAB** para obtener el peso de atención de canal $W_C \in \mathbb{R}^{C \times 1 \times 1}$.

Finalmente, W_S y W_C se multiplican, respectivamente, con el mapa de características obtenido al concatenar F y F_N para obtener el mapa de características final F_{CS}. El proceso se expresa matemáticamente mediante

- $W_C = M_C \left(\sum_{i=1}^{n} F \right)$
- $W_S = M_S(\text{Concat}([F_1, F_2, \ldots, F_n]))$
- $F_{CS} = W_S \otimes (W_C \otimes (f_{1\times1} (\text{Concat}([F, F_1, F_2, \ldots, F_n]))))$

  Donde $\otimes$ es la multiplicación elemento por elemento, y F_{CS} es el mapa de características obtenido después de la aplicación del módulo.

En este punto se procede a realizar un reentrenamiento de RetinaNet, modificando los códigos originales para poder implementar paciencia y aplicando las implementaciones que reportan el mejor desempeño en los puntos anteriores. De esta manera, se busca comparar el impacto que tiene dentro del entrenamiento e indagar su eficacia en otros modelos, asegurando que las estrategias no son eficaces solo para Y.O.L.O.

Se entrena a RetinaNet por 250 épocas y se modifican los códigos originales para implementar una paciencia propuesta de 60 épocas. Se utiliza el etiquetado 1 y se aplica una profundidad de 152 capas para la columna vertebral del modelo basado en ResNet, descrito en la subsección 3.3.2, la cual es la más grande disponible.

5.3.5. Transferencia de conocimiento

En este punto, el objetivo es contar con un modelo preentrenado en la tarea de detectar opacidades causadas por neumonía que presente un buen desempeño, para proceder con la aplicación de transferencia de conocimiento que busca extender el rendimiento a la tarea de localizar e identificar daños pulmonares causados por COVID19. El proceso propuesto se lleva a cabo de la siguiente manera.

Con el modelo preentrenado en *RSNA Pneumonia*, se realiza un reentrenamiento utilizando los parámetros aprendidos por el modelo y las estrategias de mayor impacto, esta vez usando el conjunto de entrenamiento *NIH-Chest-X-ray14*, que permitirá explorar la posibilidad de extender la capacidad de identificación para distintas clases de opacidades. Asimismo, se explora si los efectos de las implementaciones elegidas se reflejan en los resultados.

Finalmente, se realiza una última transferencia de conocimiento con el conjunto de entrenamiento *SIIM-FISABIO-RSNA COVID-19*, lo que permite evaluar el desempeño del modelo en la localización e identificación de los cuatro tipos de daños pulmonares presentes en este conjunto.

Este procedimiento se repite para la arquitectura RetinaNet, con el fin de comparar el impacto y desempeño en ambos modelos.

De esta forma, se cuenta con un marco de trabajo flexible y robusto que explora una gran parte de las posibles mejoras en cuanto a los procesos de aprendizaje profundo.

Hiperparámetro	**Valor**
hsv_h	0.5
hsv_s	0.5
hsv_v	0.5
degrees	35.5
translate	0.3
scale	0.6
shear	10.5
perspective	0
flipud	0
fliplr	0
mosaic	1.0
mixup	0.5
auto_augment	null
erasing	0.0
crop_fraction	0.5

Tabla 5.7: Aumento personalizado.

Capítulo 6

Resultados

En esta sección se abordan y discuten los resultados obtenidos al aplicar la metodología propuesta en la sección 5.3. Para el desarrollo del proyecto se realizaron 27 entrenamientos distintos, los cuales mostraron distintos resultados respecto al desempeño del modelo. Por lo tanto, en este capítulo se presenta una perspectiva general, entrando en detalle únicamente en los más relevantes para la investigación.

La información detallada de cada uno de los entrenamientos, sus gráficas, predicciones, parámetros y el repositorio del proyecto están disponibles, como se describe en el apéndice 7.

6.1. Análisis exploratorio

En esta sección se presentan los resultados obtenidos a partir de los análisis exploratorios de los conjuntos de datos descritos en la sección 5.2. Los resultados se generaron utilizando los códigos disponibles en el repositorio de GitHub descrito en el apéndice 7.

6.1.1. Conjunto RSNA Pneumonia

El conjunto de datos contiene 14,863 imágenes de rayos X anotadas con recuadros delimitadores que indican opacidades causadas por neumonía. También incluye imágenes de pacientes sanos, como se ilustra en la figura 6.1.

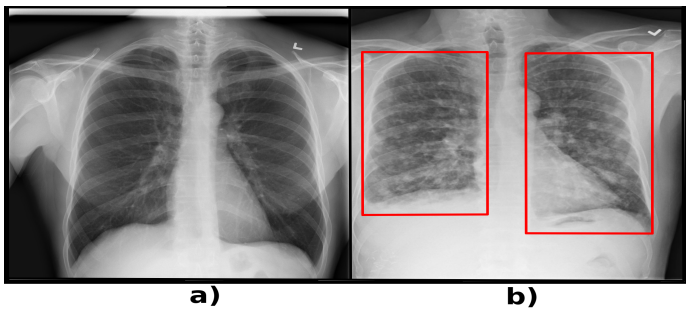


Figura 6.1: Anotaciones neumonía.

Imágenes tomadas del conjunto *RSNA Pneumonia* [22] donde: a) muestra un ejemplo de una imagen sin opacidades por neumonía, y b) una imagen con anotaciones para opacidades causadas por neumonía.

Del total de imágenes, el 40,4 % tiene anotaciones sobre la presencia de opacidades por neumonía, mientras que el 59,6 % no las contiene, lo cual representa un balance de datos aceptable. Esto se ilustra mediante una gráfica de pastel que se obtuvo mediante un conteo de clases, como se muestra en la figura 6.2.

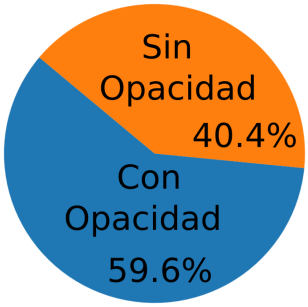


Figura 6.2: Distribución por clases.

Gráfica de pastel que ilustra el porcentaje de elementos en cada clase definida para el conjunto *RSNA Pneumonia*.

Las imágenes con presencia de opacidades causadas por neumonía cuentan con entre 1 y 4 opacidades presentes por imagen, de las cuales la mayoría abarcan un área entre el 2 % y el 10 % del área de la imagen total, como se ilustra en la figura 6.3.

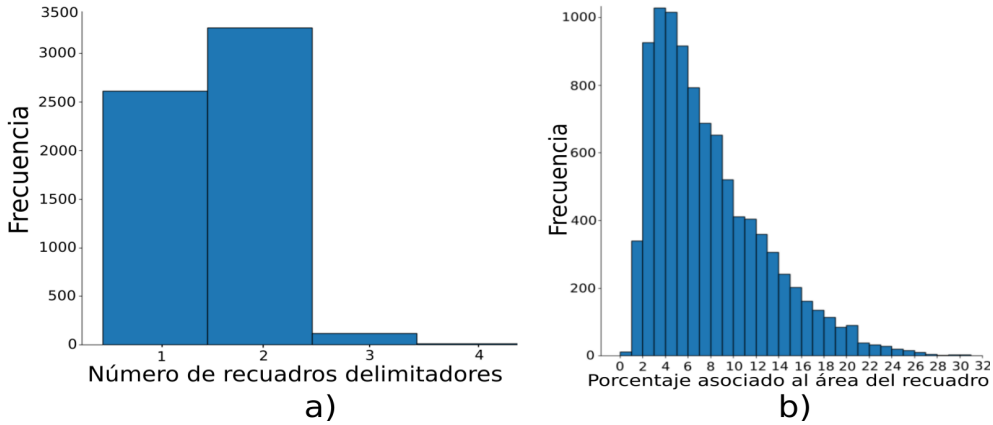


Figura 6.3: Histogramas de recuadros.

a) Histograma de recuadros por imagen. b) Histograma de porcentaje de área por recuadro.

Así se concluye con el análisis exploratorio del conjunto, haciendo énfasis en que el conjunto presenta imágenes de tamaño homogeneo, de buena resolución, y un buen balance de clases. Por lo tanto, se le considera un conjunto que contiene información que favorece el aprendizaje en la detección de opacidades generadas por neumonía.

6.1.2. Conjunto Chest X-ray14

Inicialmente se tiene que, del total de imágenes presentes en el conjunto, un 99.0 % de estas no cuentan con anotaciones de recuadros delimitadores, como se observa en la figura 6.4, esto debido a que solo 9 de las 14 clases cuentan con recuadros delimitadores aunadas a las imágenes de pacientes sin hallazgos que tampoco cuentan con recuadros.

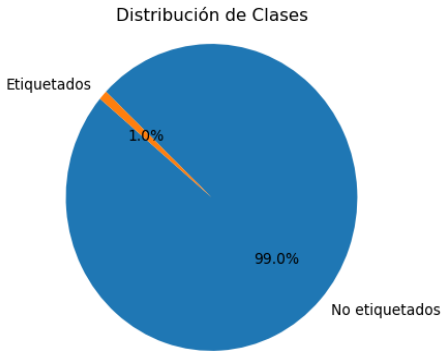


Figura 6.4: Distribución de datos.
Gráfica de pastel que ilustra el porcentaje de datos anotados.

Limitando el total a las 8 clases con anotaciones en conjunto con las imagenes sin hallazgo, se visualiza de distribución de las clases, donde se muestra un desbalance de clases por parte de las imágenes sin hallazgos con respecto al resto, dado que, en promedio, hay 122 elementos por clase. Se procede a reducir aleatoriamente el subconjunto sin hallazgos a 122 elementos para balancear las clases, como se ilustra en la figura 6.5.

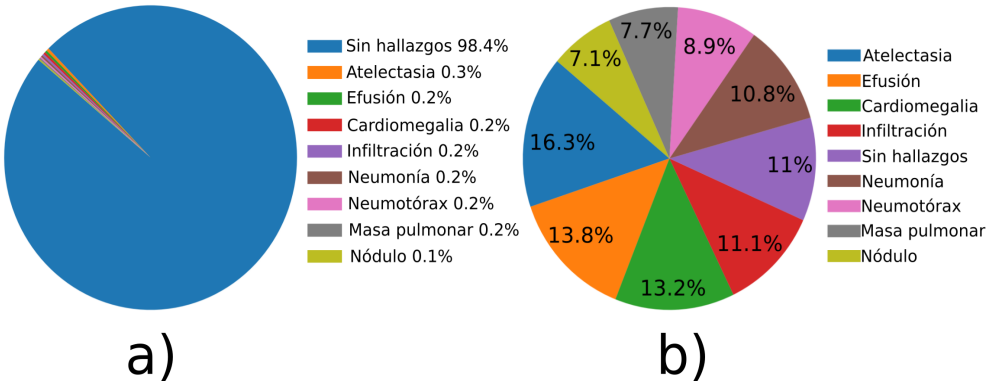


Figura 6.5: Distribución de datos por clase.
a) Porcentaje de ejemplos con desbalance de clases. b) Porcentaje de ejemplos sin desbalance de clases.

Se procede a visualizar un ejemplo de cada clase con sus respectivas anotaciones que se ilustran en la figura 6.6.

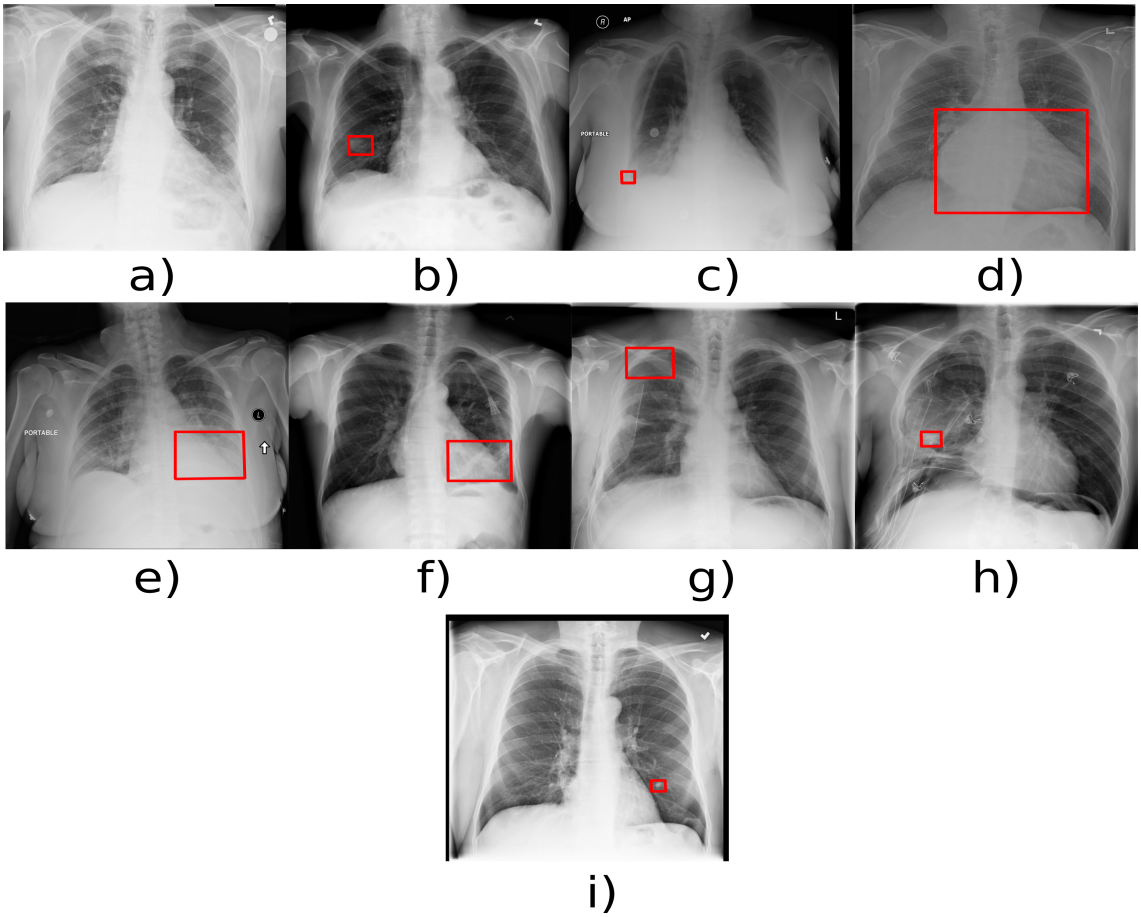


Figura 6.6: Ejemplos de datos por clases.

Ejemplos del conjunto [25] y su respectiva anotación para a) Sin hallazgos b) Atelectasia c) Efusión d) Cardiomegalia e) Infiltración f) Neumonía g) Neumotórax h) Masa pulmonar i) Nódulo.

Todas las imágenes reportan un solo recuadro delimitador, de los cuales la mayoría tienen un área entre el 1 % y el 8 % de la imagen, lo cual muestra una tendencia a recuadros pequeños como se ilustra en la figura 6.7.

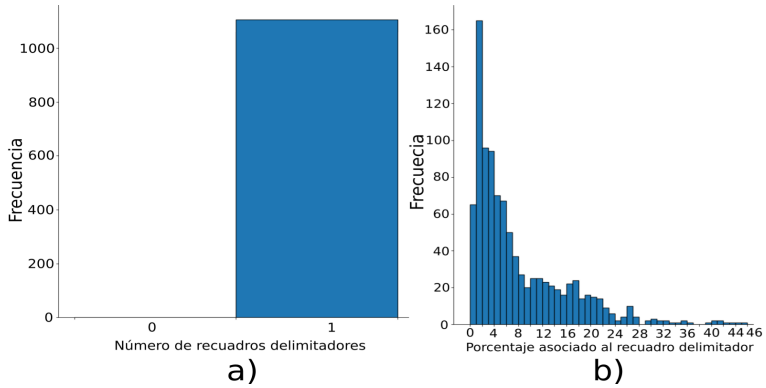


Figura 6.7: Histogramas de recuadros.

a) Histograna de recuadros por imagen. b) Histograma de porcentaje de área por recuadro.

Así se concluye con el análisis exploratorio del conjunto, haciendo énfasis en que el conjunto presenta imagenes de tamaño homogeneo y de buena resolución, pero

con una cantidad de ejemplos muy baja por clase repostada en el conjunto. Por lo tanto, se le considera un conjunto con poca información para la detección de daños pulmonares, pero con información basta para el problema de clasificación de hallazgos médicos.

6.1.3. Conjunto SIIM COVID19

El conjunto de datos cuenta con 6,334 imágenes, las cuales incluyen anotaciones de recuadro delimitador que señalan la localización de tres clases distintas de daño pulmonar y contraejemplos, detallados en la sección 5.2. En la figura 6.8 se ilustra un ejemplo de imagen por cada clase con su respectiva anotación.

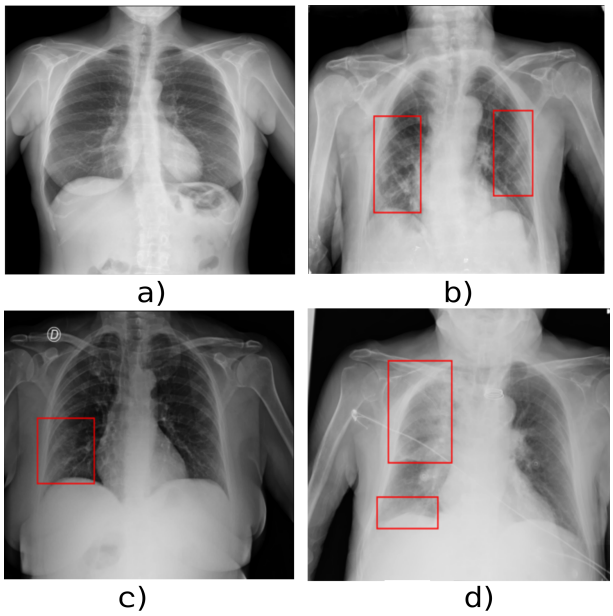


Figura 6.8: Anotaciones COVID19.

Imágenes tomadas del conjunto *SIIM COVID19* [21] con su anotación correspondiente a la clase: a) Negativo para neumonía, b) Apariencia típica, c) Apariencia indeterminada y d) Apariencia atípica.

Visualizando la distribución de clases en el conjunto, se nota un desbalance de clase para atípicos e indeterminados, como se ilustra en la gráfica de la figura 6.9.

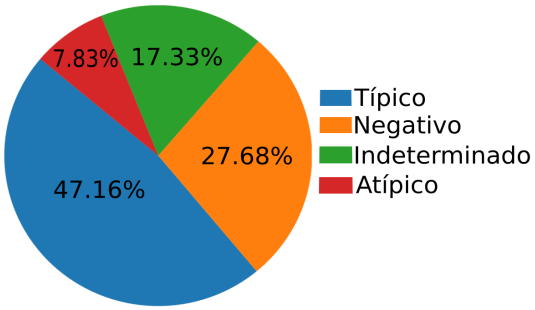


Figura 6.9: Distribución de clases.

Gráfica de pastel que ilustra el porcentaje de elementos en cada clase definida para el conjunto.

Se tienen imágenes tanto en monocromo1 como en monocromo2 (subsección 2.2.1), como se ilustra en la figura 6.10, por lo que todas las imágenes deben ser transformadas a monocromo2. Se debe tener en cuenta la inversión de colores entre negros y blancos.

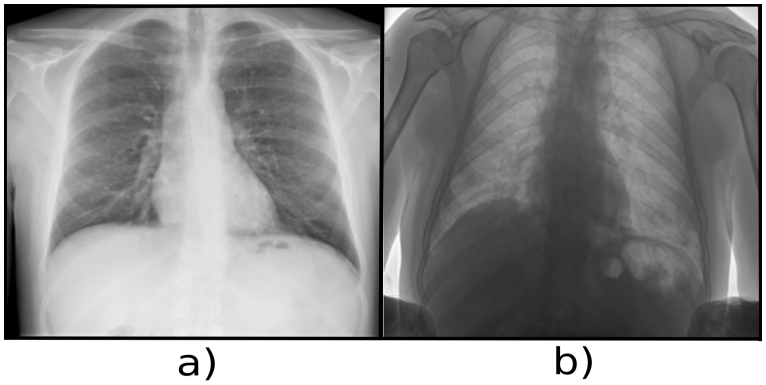


Figura 6.10: Monocromo 2 y 1.

Ejemplos de imágenes tomadas del conjunto [21]. a) Muestra una imagen en monocromo2, mientras que b) muestra una imagen en monocromo1.

Las imágenes cuentan con entre 0 y 4 recuadros delimitadores, como se puede visualizar en los siguientes histogramas, donde la mayoría tienen un área entre el 1 % y el 15 % con respecto al área de la imagen, como se muestra en la figura 6.11.

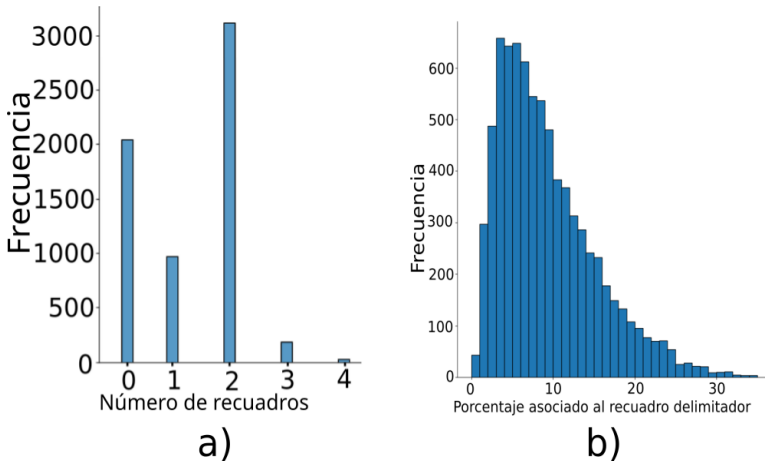


Figura 6.11: Histogramas de recuadros.

a) Histograma de recuadros por imagen. b) Histograma de porcentaje de área por recuadro.

Se procede a estimar la cantidad de imágenes sin anotaciones por clase, como se ilustra en la figura 6.12.

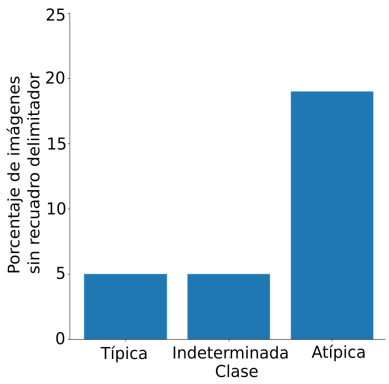


Figura 6.12: Imágenes sin recuadros delimitadores.

Gráfica que ilustra el porcentaje de imágenes sin recuadros delimitadores para cada tipo de daño pulmonar en el conjunto.

Las imágenes tienen tamaños muy variables, como se observa en la gráfica de la figura 6.13.

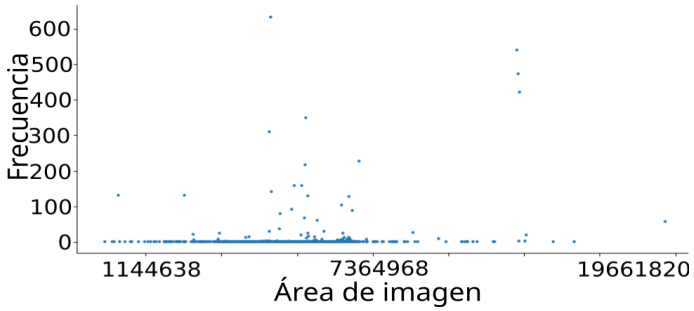


Figura 6.13: Área de las imágenes.

Gráfica que muestra el tamaño de cada imagen en el conjunto, donde el eje x representa el área de la imagen en píxeles y el eje y muestra la frecuencia de imágenes de ese tamaño.

Finalmente, este conjunto de datos presenta una gran variabilidad en cuanto a tamaño y resolución de las imágenes. Asimismo, muestra un problema de desbalanceo de clases en las categorías de atípicos e indeterminados. Además, existen muchas incoherencias en las anotaciones, ya que varias de ellas contienen recuadros que abarcan regiones fuera de las imágenes.

6.2. Hiperparámetros óptimos

Siguiendo la metodología descrita en la sección 5.3.1, la gráfica del entrenamiento básico, mostrada en la figura 6.14 a), reporta el mejor desempeño en la época 13.

Esto refleja que el optimizador SGD no está logrando encontrar un óptimo de manera eficiente, quedándose estancado en un mínimo local que no es adecuado para el rendimiento del modelo posiblemente causado por la elección inicial del lr, además de mostrar un sobreajuste (subsección 2.3.2) que se refleja en el incremento de la función de pérdida de validación, situaciones que se tendran en cuenta para los entrenamientos posteriores.

Por otro lado, para RetinaNet, se observa una convergencia en las últimas épocas del entrenamiento, como se muestra en la figura 6.14 b).

De la figura 6.14 se rescata que, para este entrenamiento inicial, RetinaNet parece presentar una mejor convergencia en las épocas finales, mientras que para Y.O.L.O, después de su rendimiento máximo, ya no logra una convergencia. Por ende, para evaluar los modelos, se utiliza el modelo de Y.O.L.O obtenido en la época 13 y para RetinaNet, el generado en la época 160.

Las matrices de confusión obtenidas para ambos modelos, mostradas en la figura 6.15, evidencian una buena capacidad para la clasificación de imágenes con opacidades y para la clasificación de contraejemplos en el conjunto de validación.

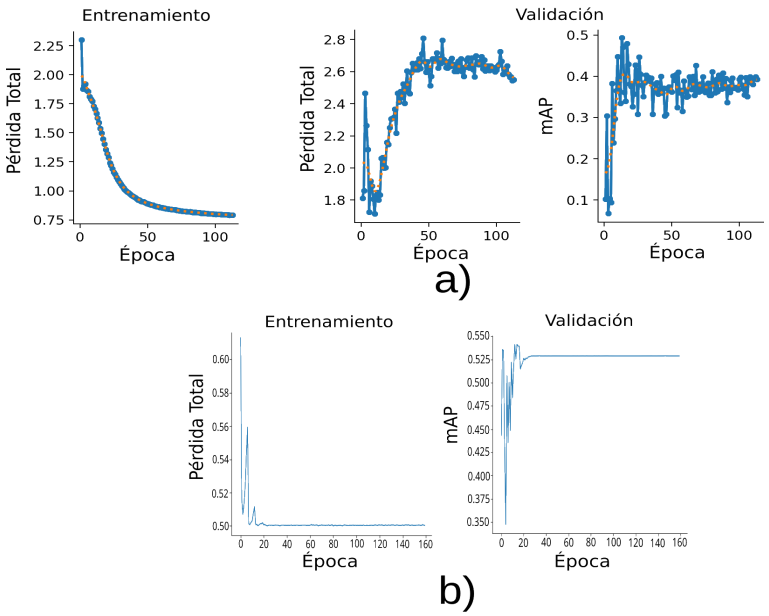


Figura 6.14: Entrenamiento básico para redes neuronales.

Gráficas del entrenamiento y validación generadas por a) Y.O.L.O [42] b) RetinaNet [43].

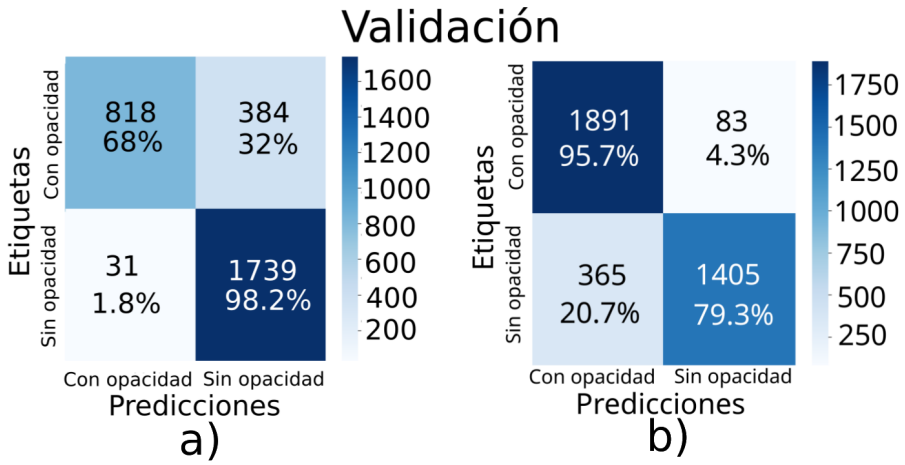


Figura 6.15: Matrices de confusión.

Matrices de confusión generadas por a) Y.O.L.O [42] b) RetinaNet [43].

Con este entrenamiento básico, Y.O.L.O reporta un mAP de 0.493 para la época 13, mientras que RetinaNet reporta un mAP de 0.541, lo que muestra una diferencia muy pequeña en cuanto a mAP para los dos entrenamientos con respecto al mismo conjunto de datos y metodología reportada en la sección 5.3. Sin embargo, respecto a las matrices de confusión, RetinaNet muestra un mejor desempeño respecto a la clasificación.

Aplicando la técnica de aumento automático, se observa un impacto positivo, alcanzando el mejor desempeño del modelo en la época 82 en la que se detuvo a causa de la condicion de paciencia de 100 épocas, como se muestra en la figura 6.16 a). Esto indica que se ayuda al optimizador en la búsqueda de los mínimos para la función de pérdida. Al observar las gráficas en la figura 6.16 a), se refleja una mejora en la convergencia del mAP y un buen decrecimiento de la pérdida. Esta implementación reporta un mAP de 0.627, lo cual representa un incremento de 0.134 unidades respecto al entrenamiento inicial, una mejora significativa.

Como se muestra en el resumen de la tabla 6.3, las técnicas de aumento aleatorio y mixto obtienen el mismo mAP, sin que se reporte ninguna diferencia entre los entrenamientos. Las gráficas obtenidas son prácticamente idénticas, lo que demuestra un impacto independiente de la técnica de aumento que se utilice.

Sin embargo, también se infiere que el uso de una técnica de aumento menos sofisticada, como el aumento personalizado, tiene un impacto negativo en comparación con el mAP de el resto de las técnicas aun cuando muestra gráficas de entrenamiento más estables, como se ilustra en la figura 6.16 b), donde se muestra un mAP de 0.568, lo que representa un decremento de 0.059 unidades con respecto al resto de las técnicas.

Aun así, se observa un incremento de 0.075 con respecto al entrenamiento básico, además de ayudar con el sobreajuste, lo que permite concluir que el aumento de datos, en cualquiera de sus implementaciones, brinda una mejora. Para los fines de esta investigación y por su impacto respecto al mAP, se decide continuar explorando la técnica de aumento automático.

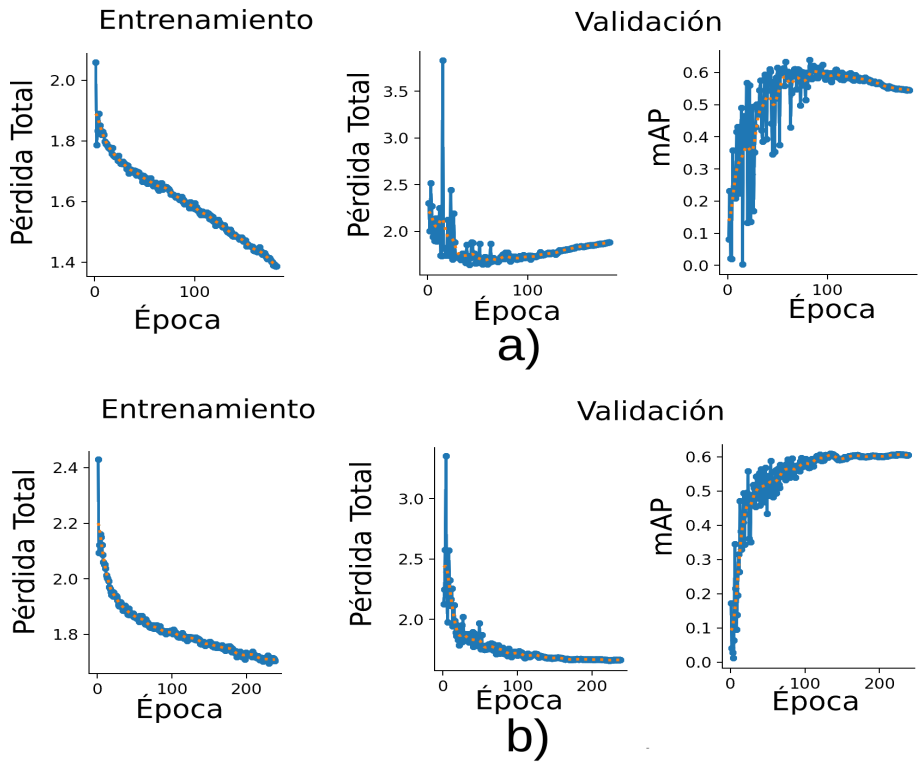


Figura 6.16: Entrenamiento Y.O.L.O+Aumento automático.

Gráficas de entrenamiento y validación generadas por Y.O.L.O [42] para a) Aumento automático b) Aumento personalizado.

Las matrices de confusión obtenidas muestran el mismo comportamiento reportado en el entrenamiento básico, como se ilustra en la figura 6.17.
Respecto a los optimizadores SGD, figura 6.16 a), y RMSProp, que tienen reglas de actualización menos sofisticadas, ambos muestran un impacto negativo en comparación con las últimas métricas obtenidas. La implementación de RMSProp reporta un mAP de 0.546, que difiere en 0.081 respecto al mAP anterior, como se ilustra en la figura 6.18 a).

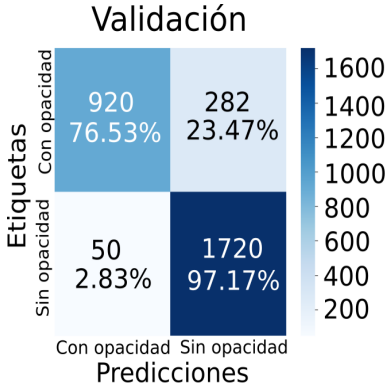


Figura 6.17: Matriz para Y.O.L.O+Aumento automático.
Matriz de confusión generada por Y.O.L.O [42].

Por otro lado, los optimizadores basados en Adam, que emplean reglas de actualización más sofisticadas, reportan en la tabla 6.3 un mAP que ronda entre 0.62 y 0.63, como se ilustra en la figura 6.18 b) para Adam. De estos, AdamW es el que ofrece el mejor resultado, con un mAP de 0.63 en la época 58, lo que refleja un incremento muy pequeño de 0.003 unidades respecto al mAP anterior de 0.627, como se muestra en la figura 6.18 c).

Las gráficas del entrenamiento con AdamW en la figura 6.18 c) muestran una disminución más rápida de la función de pérdida, mientras que el mAP exhibe una corrección respecto al ligero descenso observado en las épocas finales. En todos los casos, se reporta una reducción del sobreajuste.
La matriz de confusión para AdamW en la figura 6.19 muestra que, a pesar de haber aumentado un poco el mAP, los conteos de clasificación disminuyeron en aproximadamente un 1 %, lo cual no es problemático.
La implementación de la técnica *dropout* mantuvo intacto el modelo con un mAP reportado la tabla 6.3 de 0.63. Por otro lado, la implementación de una tasa de aprendizaje cosenoidal arrojó un mAP de 0.616, lo cual muestra un decremento de 0.014 unidades. Mientras que la especificación de clase única no presentó una mejora.

6.3. Etiquetados específicos para los datos

Como se detalla en la metodología descrita en la sección 5.3.2, se utilizan las imágenes de *RSNA Pneumonia* en su tamaño original de 1024 x 1024 píxeles, lo cual muestra un deterioro en el desempeño, arrojando un valor de mAP de 0.57. Esto representa una disminución significativa de 0.373 unidades, como se observa en la figura 6.20 a). Además, se reporta una disminución en los conteos de clasificación de aproximadamente un 10 %.

Por otro lado, al incluir dos nuevos tipos de anotaciones, se denomina **etiquetado 1** a las anotaciones que constan de un recuadro delimitador que encierra todo el contorno de la imagen, y **etiquetado 2** a las anotaciones que utilizan recuadros delimitadores para cada pulmón de manera individual.

Las gráficas del entrenamiento del modelo muestran que tanto el etiquetado 1, en la figura 6.20 b), como el etiquetado 2, en la figura 6.20 c), contribuyen a la disminución de la función de pérdida total. Además, la gráfica del mAP refleja una

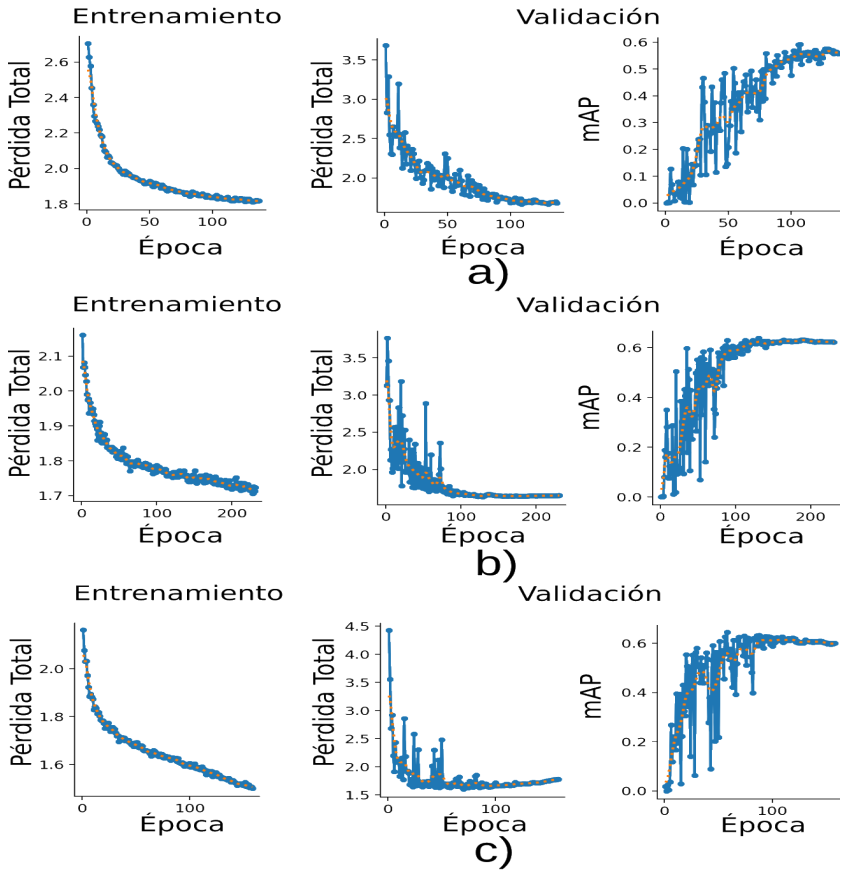


Figura 6.18: Entrenamiento Y.O.L.O con distintos optimizadores.

Gráficas de entrenamiento generadas por Y.O.L.O [42] utilizando el optimizador a) RMSProp b) Adam c) AdamW.

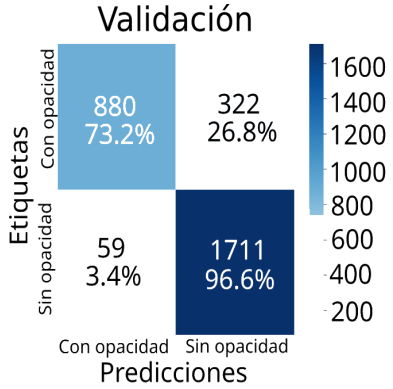


Figura 6.19: Matrices para Y.O.L.O+AdamW.

Matriz de confusión generada por Y.O.L.O [42].

mejora en la convergencia, lo que indica un efecto positivo de ambos etiquetados en la función optimizada.

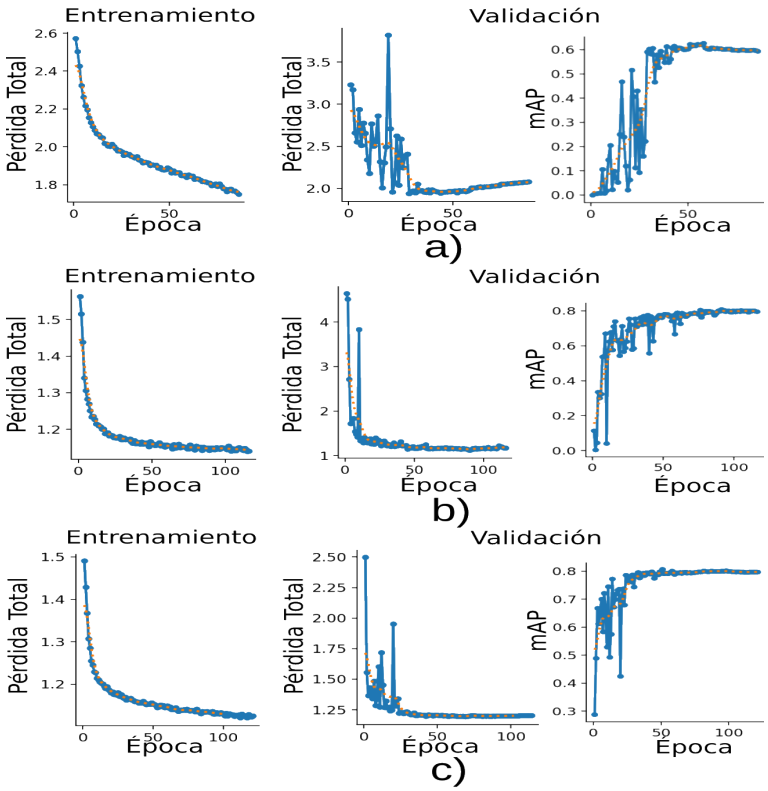


Figura 6.20: Entrenamiento Y.O.L.O+Etiqetado 1 y 2.

Gráficas generadas por Y.O.L.O [42] para a) Tamaño 1024×1024 a) Etiquetado 1 b) Etiquetado 2.

Estos entrenamientos reportan un mAP de 0.802 y 0.795 en la época 76 y 81 respectivamente para el etiquetado 1 y el etiquetado 2, lo cual refleja un buen resultado que representa un incremento respectivo de 0.172 y 0.165 unidades en comparación con el mAP reportado anteriormente de 0.63.

Las matrices de confusión en la figura 6.21 muestran que el etiquetado 1 presenta un mejor desempeño en la clasificación, con una reducción de errores para las imágenes sin opacidades respecto a los entrenamientos anteriores. También muestra mejores resultados que los obtenidos con el etiquetado 2. El etiquetado 2, en cambio, muestra un aumento en los errores de clasificación de imágenes con opacidades, lo que indica que las etiquetas para pulmones individuales introducen ruido en la detección de opacidades en el conjunto.

6.4. Definiendo profundidad de la red

Continuando con la parte de la metodología descrita en la sección 5.3.3, las gráficas del modelo *xlarge* muestran una disminución más errática en la función de pérdida en comparación con las reportadas en el entrenamiento usando el tamaño *nano*, como se observa en la figura 6.22.

De manera similar, la gráfica del mAP muestra menor estabilidad en las épocas finales del entrenamiento, evidenciando un decremento en su comportamiento. Para este entrenamiento, fue necesario definir los hiperparámetros *momentum*=0.001, *warmup_momentum*=0.001, *lr0*=0.001, *lrf*=0.001 y *patience*=50.

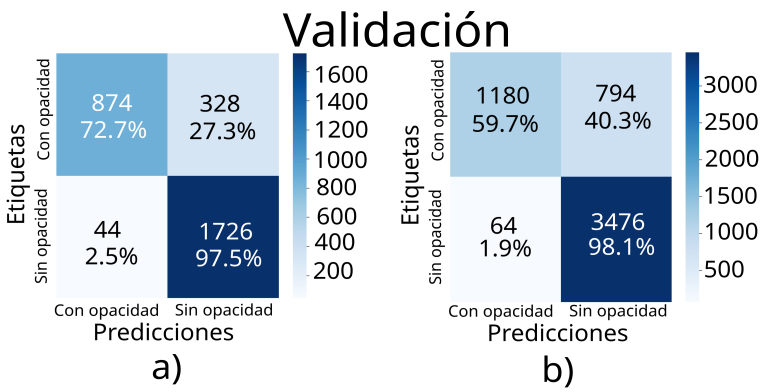


Figura 6.21: Matrices de confusión para Y.O.L.O + Etiquetado 1 y 2.

Matrices de confusión generadas por Y.O.L.O [42] para a) Etiquetado 1 b) Etiquetado 2.

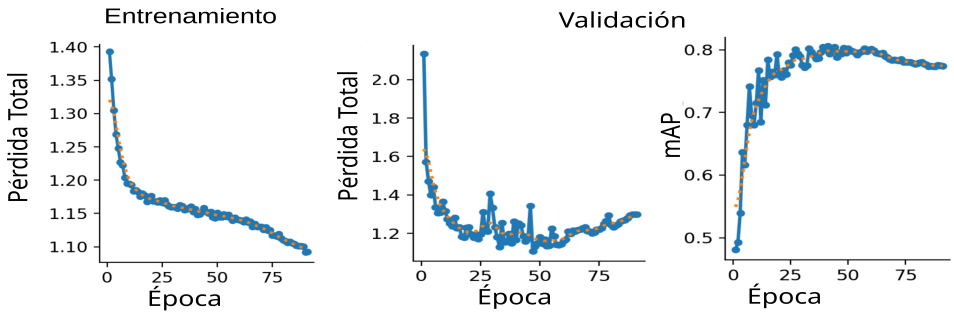


Figura 6.22: Entrenamiento Y.O.L.O *xlarge*.

Gráficas generadas por Y.O.L.O [42].

La matriz de confusión en la figura 6.23 muestra una buena capacidad de generalización en el conjunto de prueba, reflejando un incremento aproximado del 10 % en el conteo de errores respecto al uso del tamaño *nano*.

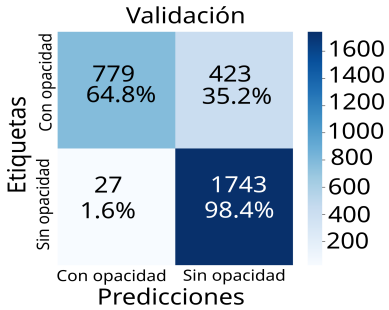


Figura 6.23: Matrices para Y.O.L.O *xlarge*.

Matriz de confusión generada por Y.O.L.O [42].

Como se reporta en la tabla 6.3, los modelos de profundidad *medium* a *xlarge* logran un mAP entre 0.78 y 0.79, lo cual muestra una reducción en el desempeño del modelo. Dado que la profundidad *xlarge* alcanza un mAP de 0.803, lo cual representa una mejora de 0.001 unidades con respecto al logrado por la profundidad *nano*, en la sección 6.5 se explora si la implementación de módulos de atención logra una mejora significativa en el modelo. De lo contrario, se continuará con la profundidad *nano*, ya que el poco incremento en el mAP no compensa el aumento en los recursos computacionales y de tiempo que requiere la profundidad *xlarge*.

6.5. Modificación de módulos de atención

Implementando la metodología de la sección 5.3.4, se observa que Y.O.L.O ya cuenta con un módulo de atención con topología piramidal de capas convolucionales. Por lo tanto, se procede a modificar dicho módulo, agregando capas de convolución con los filtros propuestos en el artículo de Ma, X., et al. [7].

Las gráficas de entrenamiento en la figura 6.24 del modelo muestran un comportamiento idéntico al obtenido para Y.O.L.O *xlarge* sin modificar. El entrenamiento presenta exactamente las mismas matrices de confusión, así como el mismo mAP de 0.803, por lo que se concluye que modificar las capas de convolución en la capa de atención no aporta ninguna mejora. En consecuencia, se procede a continuar con la arquitectura original.

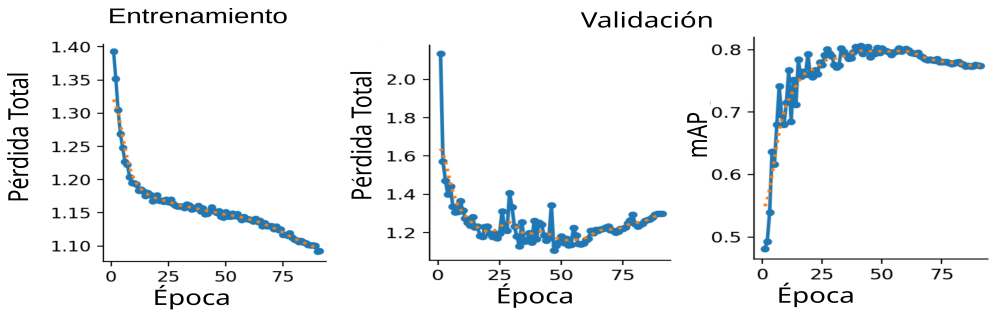


Figura 6.24: Entrenamiento Y.O.L.O+Atención.
Gráficas generadas por Y.O.L.O [42].

6.6. Prueba de eficacia de las estrategias propuestas

Como se indica en la última parte de la metodología de la sección 5.3.4 de metodología, para evaluar el impacto de las estrategias adoptadas hasta este momento, se procede a realizar un nuevo entrenamiento del modelo RetinaNet y observar su efecto sobre las métricas obtenidas en el entrenamiento básico inicial.

Dado que RetinaNet implementa un extractor de características con topología piramidal de capas convolucionales, no se realizan cambios en el módulo de atención y se mantiene el optimizador SGD.

Las gráficas de entrenamiento y validación del modelo en la figura 6.25 a) muestran una convergencia con oscilaciones menos prolongadas respecto al entrenamiento básico de RetinaNet ilustrado en la figura 6.25 b).

El mejor rendimiento se reporta en la época 6, por lo que se procede a evaluar el modelo en la época 66 definida por la paciencia, mostrando una mejor convergencia para la función de pérdida y el mAP.
La matriz de confusión en la figura 6.26 a) muestra una disminución aceptable en los aciertos para la clasificación de imágenes con opacidades, pero reflejan una mejora significativa en el caso de la ausencia de opacidades respecto al entrenamiento básico ilustrado en la figura 6.26 b).

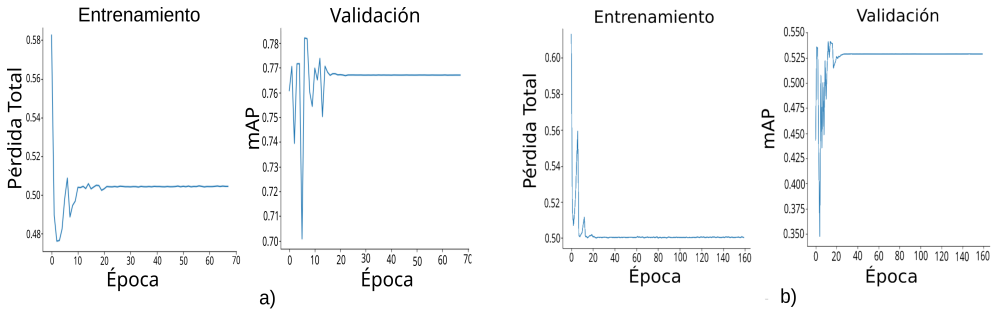


Figura 6.25: Entrenamiento Retina.

Gráficas generadas por RetinaNet [43] de a) Retina+Estrategias b)Entrenamiento básico.

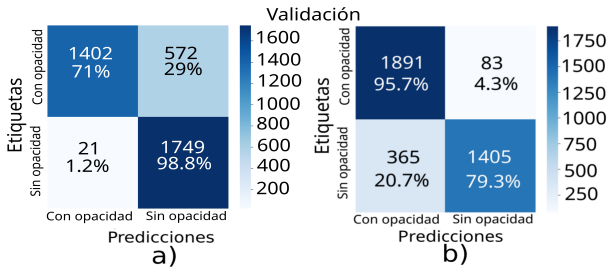


Figura 6.26: Matrices de confusión para RetinaNet.

Matrices de confusión generada por RetinaNet [43] para: a) Implementación de estrategias, b) Entrenamiento básico.

Este entrenamiento reporta un mAP de 0.767, lo que representa un incremento de 0.226 unidades. Esto indica que, la implementación de las estrategias exploradas tiene un impacto positivo en los modelos y sus entrenamientos.

6.7. Transferencia de conocimiento

Siguiendo lo descrito en la metodología de la sección 5.3.5, se implementan las estrategias de mejor desempeño para Y.O.L.O en su versión *nano*.

Las gráficas del entrenamiento del modelo en la figura 6.27 muestran un comportamiento errático, caracterizado por oscilaciones durante todas las épocas para la función de pérdida y el mAP, respectivamente, así como un claro comportamiento de sobreajuste.

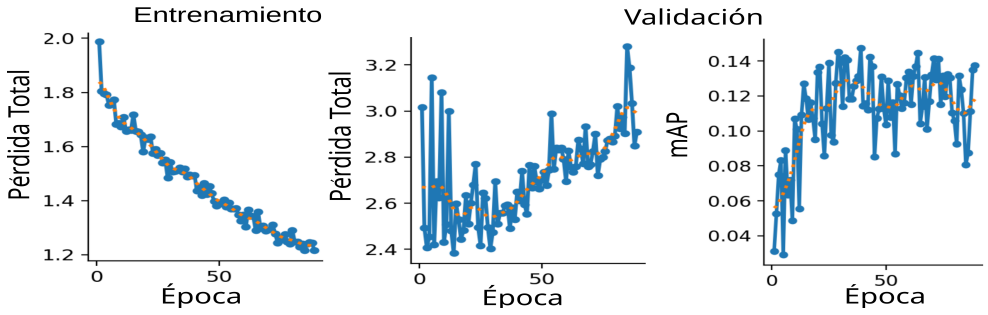


Figura 6.27: Entrenamiento Y.O.L.O+ *Chest Xray14*.

Gráficas generadas por Y.O.L.O [42].

La matriz de confusión en la figura 6.28 refleja una nula capacidad de identificación

en el conjunto de validación. Esto indica que el modelo no tiene la capacidad de generalizar ni de aprender las características del conjunto para realizar una identificación correcta, debido a la escasa cantidad de ejemplos disponibles para entrenar el modelo.

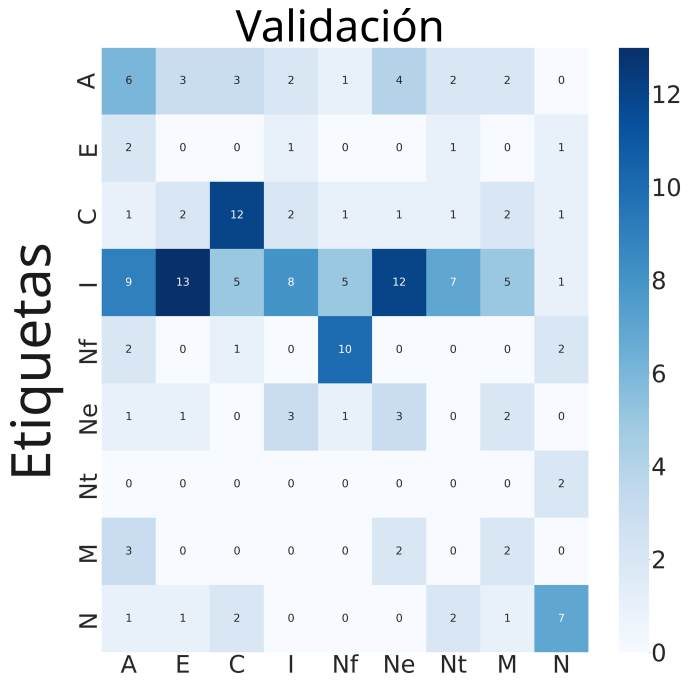


Figura 6.28: Matrices para Y.O.L.O+ *Chest Xray14*.
Matriz de confusión generada por Y.O.L.O [42].

Para la matriz en la figura 6.28, se adoptó la siguiente nomenclatura.

A: Atelectasia, **E:** Efusión, **C:** Cardiomegalia, **I:** Infiltración, **Nf:** Sin hallazgos, **Ne:** Neumonía, **Nt:** Neumotórax, **M:** Masa y **N:** Nódulo.

Se observa que el modelo no extrapola el conocimiento adquirido, por lo que se procede a realizar una transferencia directa de conocimiento con el conjunto *SIIM COVID-19*, tanto para Y.O.L.O como para RetinaNet.

Las gráficas de entrenamiento y validación del modelo en la figura 6.29 muestran que, en el caso de Y.O.L.O, la función de pérdida presenta un decrecimiento oscilante, mientras que el mAP muestra un crecimiento con oscilaciones significativas durante todo el entrenamiento.

Para RetinaNet, la función de pérdida muestra un decremento con ligeras variaciones erráticas al inicio, pero con una estabilización al final, mientras que el mAP presenta un crecimiento oscilatorio que también se estabiliza en las épocas finales.

Por otro lado, las matrices de confusión de ambos modelos en la figura 6.30 muestran conteos muy similares para la identificación y reportan una clara dificultad para identificar lesiones indeterminadas.

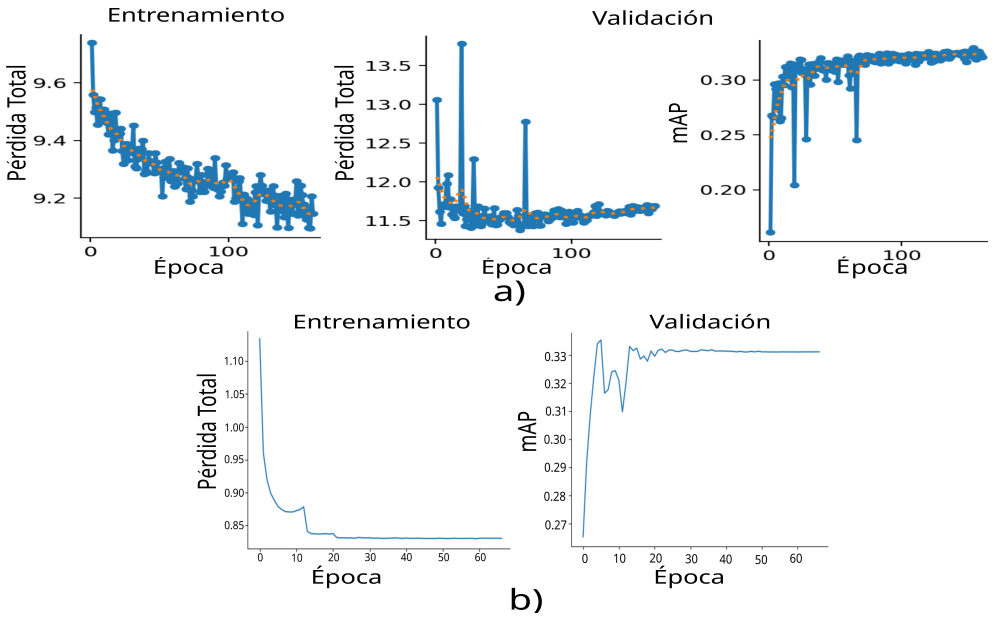


Figura 6.29: Entrenamientos *SIIM COVID19*.
Gráficas generadas por a) Y.O.L.O [42] y b) RetinaNet [43].

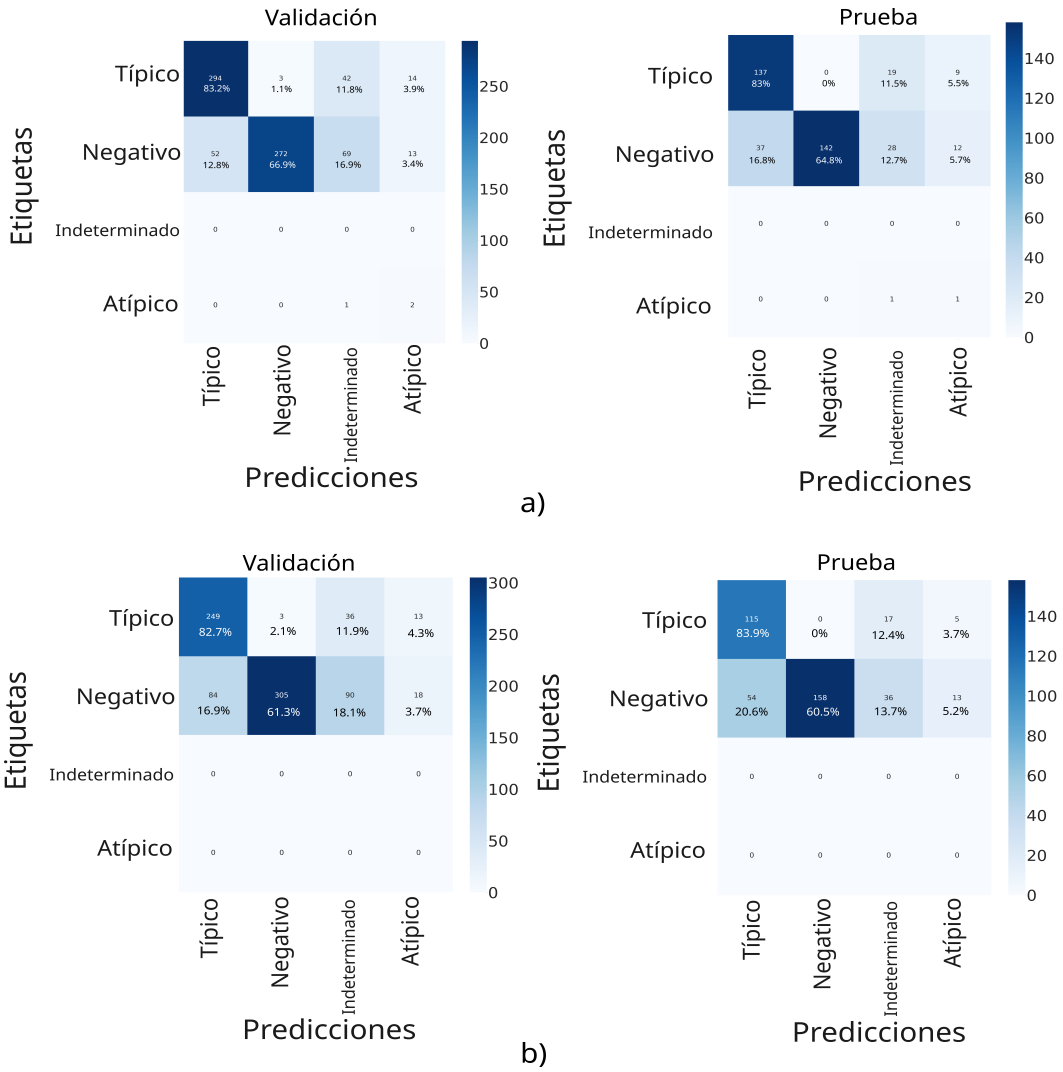


Figura 6.30: Matrices para *SIIM COVID19*.
Matrices de confusión generadas por a) Y.O.L.O [42] y b) RetinaNet [43].

Los mAPs reportados tuvieron un valor de 0.325 para Y.O.L.O y 0.331 para RetinaNet, lo que indica una clara disminución de 0.196 unidades en el mAP con respecto al mejor obtenido para *RSNA Pneumonia*, que fue de 0.802. Aun así, se muestra un buen desempeño para 2 de las 4 clases que conforman el conjunto.

Respecto a las predicciones de los modelos, se puede observar un comportamiento muy similar en cuanto a la posición y tamaño de los recuadros delimitadores. Y.O.L.O muestra un mejor desempeño en los casos donde hay presencia de múltiples recuadros en la imagen, como se ilustra en la figura 6.31 a). En los casos en que RetinaNet localiza solo un recuadro, lo hace correctamente. Además, se destaca la nula capacidad de identificar daños atípicos e indeterminados, como se ilustra en la figura 6.31 b).

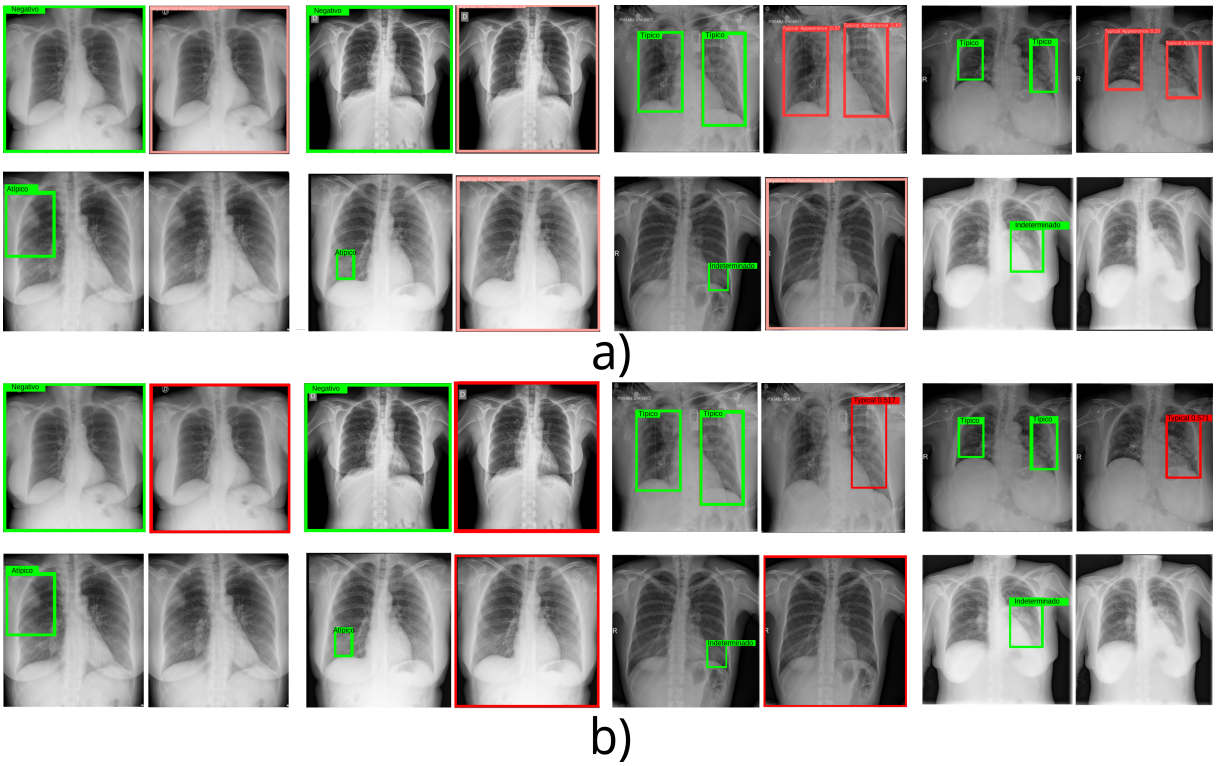


Figura 6.31: Predicciones para *SIIM COVID19*.

En la imagen, el primer renglón muestra, de dos en dos, ejemplos de imágenes con presencia de opacidades, mientras que el segundo renglón presenta imágenes de personas sanas. En (verde) se muestran las anotaciones manuales, y en (rojo/rosa) las predicciones generadas por a) Y.O.L.O [42] y b) RetinaNet [43].

Finalmente se proporciona la tabla 6.1[1] que muestra la confiugración final utilizada para la tranferencia de conocimiento en Y.O.L.O.

Configuración final Y.O.L.O	
Parámetro	**valor**
epochs	350
patience	50
batch	121
imgsz	640
optimizer	AdamW
cos_ lr	false
dropout	0.0
augment	true
lr0	0.001
cls	0.5
dfl	1.5
hsv_ h	0.015
hsv_ s	0.7
hsv_ v	0.4
degrees	0.0
translate	0.1
scale	0.5
shear	0.0
perspective	0.0
flipud	0.0
fliplr	0.5
bgr	0.0
mosaic	1.0
mixup	0.0
copy_ paste	0.0
auto_ augment	autoaugment
erasing	0.4
crop_ fraction	1.0

Tabla 6.1: Configuración final para Y.O.L.O.

[1]Los detalles de entrenamiento y parámetros para cada uno de los entrenamientos realizados se pueden encontrar en el archivo args.yml y train.txt en la respectiva carpeta de la siguiente liga *https://drive.google.com/drive/folders/1JVR-FKDxJcaKLuTDaTM2A9S_6f6m2J4A?usp=drive_link*

6.8. Resumen de resultados

En las tablas 6.2 y 6.3 se presenta un resumen de los mAPs obtenidos en cada uno de los entrenamientos realizados durante el desarrollo de esta investigación. Se destacan en azul los resultados más significativos. Los detalles y las gráficas correspondientes de cada entrenamiento pueden ser consultados a través de los enlaces proporcionados en el apéndice 7 de este trabajo.

Resumen de mAPs por entrenamiento para RetinaNet			
Entrenamiento	**mAP**	**Tamaño lote**	**Tiempo entrenamiento**
16. Básico	0.5413	123	6.815 horas
25. Implementación de estrategias	0.7670	123	5.712 horas
27. Transferencia de conocimiento	0.3310	123	6.193 horas

Tabla 6.2: Resumen de mAPs obtenidos para RetinaNet.

Resumen de mAPs por entrenamiento para Y.O.L.O			
Entrenamiento	**mAP**	**Tamaño lote**	**Tiempo entrenamiento**
1. Básico	0.493	125	1.196 horas
2. Implementación RandAugment	0.627	123	1.985 horas
3. Implementación AutoAugment	0.627	123	1.907 horas
4. Implementación AugMix	0.627	123	1.916 horas
5. Aumento personalizado	0.568	123	3.361 horas
6. Optimizador Adam	0.623	123	2.498 horas
7. Optimizador Adamax	0.620	123	2.488 horas
8. Optimizador AdamW	0.630	123	1.714 horas
9. Optimizador NAdam	0.619	123	2.193 horas
10. Optimizador RAdam	0.625	123	2.221 horas
11. Optimizador RMSProp	0.546	123	1.466 horas
12. Clase única	0.630	123	1.642 horas
13. Tasa cosenoidal	0.616	123	1.716 horas
14. Dropout	0.630	123	1.679 horas
15. Modificación DFL	0.619	123	1.750 horas
17. Imagen 1024x1024	0.570	48	2.020 horas
18. Etiquetado 1	0.802	121	1.371 horas
19. Etiquetado 2	0.795	14	3.179 horas
20. Small	0.793	66	2.608 horas
21. Medium	0.780	17	2.839 horas
22. Large	0.797	13	4.402 horas
23. Xtralarge	0.803	17	7.491 horas
24. Modificación de arquitectura	0.803	17	7.554 horas
26. Transferencia de conocimiento	0.325	121	0.789 horas

Tabla 6.3: Resumen de mAPs obtenidos para Y.O.L.O.

6.8.1. Discusión sobre resultados

Se trabajó con tres conjuntos de datos, de los cuales el análisis exploratorio reveló que el conjunto con mayor calidad en cuanto a la información es *RSNA Pneumonia*, ya que presenta una excelente calidad de imagen y homogeneidad tanto en sus anotaciones como en sus formatos.

El conjunto *Chest Xray14* mostró que, aunque cuenta con un gran número de imágenes, solo una pequeña cantidad de ellas tiene anotaciones para la detección de hallazgos médicos. Esto genera un conjunto con muy pocos ejemplos por clase, aunque de buena calidad de imagen.

Por último, el conjunto *SIIM-FISABIO-RSNA COVID-19* resultó ser homogéneo en varios aspectos: las imágenes tenían tamaños muy variados, algunas estaban representadas en rangos ascendentes de tonos de gris y otras en descendentes. La calidad de las imágenes también era muy variable. Además, muchas de las anotaciones presentaban problemas, ya que excedían los límites de las imágenes, lo que evidencia la presencia de anotaciones erróneas y genera dudas sobre la calidad del resto de las anotaciones. Esto es comprensible, dado el contexto de emergencia global en el que se construyó el conjunto.

Los entrenamientos realizados revelan información interesante sobre el impacto de las decisiones tomadas en el rendimiento de los modelos. Los primeros 15 entrenamientos permitieron explorar los mejores hiperparámetros para Y.O.L.O en su versión *nano*. Se determinó que era necesario aplicar un aumento de datos automático y utilizar el optimizador AdamW para Y.O.L.O, ya que brinda mayor estabilidad al modelo. Por otro lado, para RetinaNet se optó por el uso de SGD, dado que la implementación de AdamW en este caso generaba valores no numéricos desde las primeras épocas del entrenamiento, mostrando inestabilidad numérica.

En los entrenamientos del 17 al 19, se exploraron modificaciones en los datos de entrenamiento. El entrenamiento 17 mostró que el aumento en el tamaño de las imágenes genera una disminución en el desempeño del modelo debido al incremento en la complejidad de los datos. Por otro lado, el entrenamiento 18 ilustró cómo las anotaciones propuestas permiten una mejor interpretabilidad en relación con las métricas y predicciones.

En los entrenamientos del 20 al 23 se exploró la profundidad del modelo, observándose que, para las profundidades *small*, *medium* y *large*, el mAP reportado se mantiene entre 0.78 y 0.79. Esto sugiere que, a mayor tamaño, el modelo presenta una ligera disminución en su rendimiento, lo cual se atribuye al incremento en el número de parámetros a entrenar. Por otro lado, el tamaño *xlarge* mostró una mejora mínima de 0.001, lo cual no es significativo ni conveniente en comparación con el tiempo y los recursos computacionales necesarios para entrenar un modelo tan grande. Por ello, se consideró conveniente continuar con el tamaño *small*.

El entrenamiento 24 mostró que la modificación de la capa de atención en la arquitectura de Y.O.L.O no tuvo ningún efecto en el rendimiento, dado que el modelo ya implementaba una capa de atención con diferentes capas de convolución. Por esta razón, se decidió no realizar modificaciones en RetinaNet, ya que esta red también cuenta con un modelo de atención piramidal.

El entrenamiento 25 arrojó resultados importantes, ya que demostró que las estrategias y decisiones tomadas para el entrenamiento con Y.O.L.O tienen un impacto positivo similar en el desempeño de RetinaNet. Esto confirma la eficacia de la metodología propuesta, reportando un aumento en el mAP de 0.541 a 0.767, lo que representa un incremento significativo.

Finalmente, los entrenamientos 26 y 27, relativos a la transferencia de conocimiento, reportaron que, haciendo uso del modelo preentrenado y el conjunto *ChestX-ray14*, el buen desempeño reportado por el modelo baja debido a la baja cantidad de ejemplos por clases en el conjunto, lo que reafirma que no basta solo con tener una buena calidad de imagen, también es necesario una cantidad de imágenes representativa en el conjunto. Dada esta situación, se procedió a reentrenar el modelo directamente con el conjunto *SIIM-FISABIO-RSNA COVID19* y de la misma forma para el entrenamiento 27 de RetinaNet.

Dado el desempeño en el aprendizaje obtenido por el modelo del conjunto *RSNA Pneumonia*, se observó que, por una parte, el modelo extiende su capacidad de detección de opacidades, mostrando un buen desempeño para la localización e identificación de lesiones causadas por neumonía típica de COVID19 y negativos para neumonía por COVID19, mientras que para las lesiones atípicas e indeterminadas se ve una baja capacidad de aprendizaje debido a la calidad de imágenes y la baja representación de las clases.

Aun con estas adversidades y tomando el mAP de las dos clases más representativas, se obtiene un mAP total de 0.606 y 0.622 para cada modelo, respectivamente. Estos resultados son satisfactorios para la presente investigación, ya que tanto estudios previos como la competencia *Kaggle* reportan un mAP total que ronda entre 0.5951 y 0.635 para las cuatro clases, lo que indica una baja aportación equitativa de cada clase al mAP total. Esta pequeña diferencia podría deberse a que en dichos estudios se implementaron métodos para contrarrestar el desbalance de clases, mientras que en la presente investigación no se utilizaron, dado que el enfoque se centró en medir la influencia exclusivamente de las estrategias aquí propuestas.

Capítulo 7

Conclusiones y perspectivas

El presente trabajo documenta estrategias que resultan determinantes para el correcto desempeño y rendimiento de los modelos Y.O.L.O y RetinaNet en la tarea de localizar e identificar daños relacionados con la enfermedad infecciosa COVID19. A pesar de las dificultades y deficiencias relacionadas con los datos de entrenamiento, se logró un incremento en el mAP, pasando de 0.493 a 0.8 en el mejor conjunto de entrenamiento.

Las estrategias exploradas presentaron robustez y mostraron efectividad en la toma de decisiones para el entrenamiento de modelos, lo que puede aplicarse como metodología de trabajo para cualquier otro modelo, conjunto de datos o tarea. Además, estas estrategias resaltan características en los conjuntos de entrenamiento que pueden contribuir a un buen desempeño en la provisión de soluciones para tareas que impliquen visión computacional en la localización e identificación de objetos.

Esto ejemplifica el equilibrio necesario entre la calidad de los datos y un modelo robusto. Por otro lado, se deja abierta la posibilidad de explorar futuras soluciones que, en conjunto o en sustitución de las aquí propuestas, brinden un mejor rendimiento en modelos de aprendizaje profundo.

El trabajo sugiere diversas opciones para explorar en el futuro, como evaluar el impacto que tiene la aplicación conjunta de las estrategias con técnicas que aborden el desbalance de clases presente en el conjunto *SIIM-FISABIO-RSNA COVID19*. Esto se puede lograr mediante una función de pérdida que penalice los errores en clases poco representadas, utilizando técnicas como el sobremuestreo o el aumento de datos para estas clases particulares.

Por otro lado, dada la baja cantidad de ejemplos para algunas clases de *ChestX-ray14*, podría buscarse o construirse un nuevo conjunto de datos que contenga ejemplos balanceados de distintos tipos de opacidades en radiografías de pulmón. Esto permitiría generar un aprendizaje que identifique de forma eficiente diversos tipos de opacidades e investigar el impacto de aplicar transferencia de conocimiento.

Otro esfuerzo valioso para la comunidad científica sería generar un nuevo conjunto de datos con anotaciones similares a las propuestas para *SIIM-FISABIO-RSNA COVID19*, con el objetivo de crear un conjunto que comparta una calidad similar a la proporcionada por *RSNA Pneumonia*. Este nuevo conjunto de datos podría servir como base para probar nuevos modelos y futuros desarrollos.

Apéndice: Repositorio GitHub

Los códigos utilizados para el desarrollo de la presente investigación se encuentran en el repositorio público de GitHub[1], donde se brindan los detalles necesarios para replicar los resultados.

Dicho repositorio contiene una carpeta principal bajo el nombre de COVID que, en su interior, contiene datos descritos por la siguiente estructura de subcarpetas.

- **Anotaciones:** Carpeta que contiene las distintas anotaciones utilizadas para entrenar los modelos de aprendizaje profundo. Para cada conjunto de datos, existe una carpeta que contiene las anotaciones únicamente para RetinaNet, ya que las anotaciones para Y.O.L.O se generan a partir de los archivos contenidos en la carpeta Datos y los códigos contenidos en la carpeta Códigos_Datos, ambos presentes en este repositorio. Cada comprimido contiene un archivo de texto con la descripción de las anotaciones correspondientes.
- **Códigos_Datos:** Carpeta que contiene tres cuadernos de Jupyter Lab, cada uno con códigos y funciones utilizadas para generar las anotaciones contenidas en la carpeta Anotaciones y el procesamiento de datos para cada conjunto.
- **Datos:** Contiene archivos de texto respectivos a cada uno de los conjuntos de datos utilizados, donde se resume de manera estructurada la información relevante sobre cada conjunto.
- **Exploratorios:** Contiene tres cuadernos de Jupyter Lab, uno por cada conjunto de datos utilizado. Estos cuadernos contienen los códigos y funciones utilizados para realizar los análisis exploratorios reportados.
- **env.yml:** Configuración de ambiente virtual de Anaconda que contiene todas las librerías necesarias para los análisis exploratorios, visualización, preprocesamiento de datos y manipulación de etiquetados.
- **Entrenamientos:** Los pesos de los modelos entrenados descritos en la tesis, así como las gráficas e información de entrenamiento completa, están disponibles en la carpeta de Drive[2].
- **Códigos_Modificados:** Contiene dos carpetas, una para cada modelo:

  Para el caso de Y.O.L.O, se incluye el código con las modificaciones en el módulo de atención, así como el cuaderno de Jupyter Lab utilizado en el

[1]Disponible en la liga https://github.com/JairMathAI/COVID

[2]Disponible en la liga https://drive.google.com/drive/folders/1JVR-FKDxJcaKLuTDaTM2A9S_6f6m2J4A?usp=drive_link

entrenamiento que contempla las modificaciones para la generación de matrices.

Para RetinaNet, se incluyen los códigos:

AumentoSIIM.ipynb: Código que permite hacer aumento de datos para RetinaNet.

ConfuseMatrix.py: Código desarrollado para la generación de matrices de confusión para RetinaNet.

trainModif.py: Código modificado de train.py disponible en el GitHub del modelo RetinaNet, que implementa el uso de paciencia, guardado del modelo con mejor desempeño, la generación de gráficas de pérdida total y mAP, así como la transferencia de conocimiento.

modelModif.py: Código modificado de model.py disponible en el GitHub del modelo RetinaNet, que implementa un bosquejo del módulo de atención no utilizado, pero se añade por si se quisiera explorar y/o mejorar.

visualize_single_image.py: Código modificado de visualize_single_image.py disponible en el GitHub del modelo RetinaNet, al cual se le realizaron correcciones ya que al ejecutarse este presenta fallos.

Bibliografía y referencias

[1] Kumar, M., et al. (2022). Detection of COVID-19 using deep learning techniques and cost effectiveness evaluation: A survey. *Frontiers in Artificial Intelligence*, *5*. https://doi.org/10.3389/frai.2022.912022

[2] Hijikata, Y., et al. (2010). Dyspnoea, fever, patchy ground-glass opacities and intermittent severe epigastralgia. *Thorax*, *65*, 890–890. https://doi.org/10.1136/thx.2010.136655

[3] Lin, T.-Y., et al. (2014). Microsoft COCO: Common objects in context. *arXiv*. https://doi.org/10.48550/ARXIV.1405.0312

[4] Tan, M., et al. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. In K. Chaudhuri & R. Salakhutdinov (Eds.), *Proceedings of the 36th International Conference on Machine Learning* (pp. 6105–6114). Proceedings of Machine Learning Research.

[5] Marcomini, K. D., et al. (2022). A deep learning approach for COVID-19 screening and localization on chest X-ray images. *Proc. SPIE 12033, Medical Imaging 2022: Computer-Aided Diagnosis, 1203327*. https://doi.org/10.1117/12.2613177

[6] Polat, H., et al. (2021). Automatic detection and localization of COVID-19 pneumonia using axial computed tomography images and deep convolutional neural networks. *International Journal of Imaging Systems and Technology*, *31*(2), 509-524. https://doi.org/10.1002/ima.22558

[7] Ma, X., et al. (2021). COVID-19 lesion discrimination and localization network based on multi-receptive field attention module on CT images. *Optik*, *241*. https://doi.org/10.1016/j.ijleo.2021.167100

[8] Carandini, M., et al. (2006). What simple and complex cells compute. *Journal of Physiology*, 463-466. https://doi.org/10.1113/jphysiol.2006.118976

[9] LeCun, Y., et al. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278-2324. https://doi.org/10.1109/5.726791

[10] Vaswani, A., et al. (2017). Attention is all you need. *arXiv*. https://doi.org/10.48550/ARXIV.1706.03762

[11] Dosovitskiy, A., et al. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv*. https://doi.org/10.48550/ARXIV.2010.11929

[12] Tan, M., et al. (2019). EfficientDet: Scalable and efficient object detection. *arXiv*. https://doi.org/10.48550/ARXIV.1911.09070

[13] Redmon, J., et al. (2015). You only look once: Unified, real-time object detection. *arXiv*. https://doi.org/10.48550/ARXIV.1506.02640

[14] Manliguez, C. (2016). Generalized confusion matrix for multiple classes. *ResearchGate*. https://doi.org/10.13140/RG.2.2.31150.51523

[15] Gomes, J. C., et al. (2022). IKONOS: An intelligent tool to support diagnosis of COVID-19 by texture analysis of X-ray images. *Research on Biomedical Engineering*, *38*(1), 15–28. https://doi.org/10.1007/s42600-020-00091-7

[16] Wang, D., et al. (2020). An efficient mixture of deep and machine learning models for COVID-19 diagnosis in chest X-ray images. *PLOS ONE*, *15*(11), e0242535. https://doi.org/10.1371/journal.pone.0242535

[17] Shen, D., et al. (2017). Deep learning in medical image analysis. *Annual Review of Biomedical Engineering*. https://doi.org/10.1146/annurev-bioeng-071516-044442

[18] Kuchana, V., et al. (2021). AI aiding in diagnosing, tracking recovery of COVID-19 using deep learning on chest CT scans. *Multimedia Tools and Applications*, *80*, 9161–9175. https://doi.org/10.1007/s11042-020-10010-8

[19] Yang, D., et al. (2021). Detection and analysis of COVID-19 in medical images using deep learning techniques. *Scientific Reports*. https://doi.org/10.1038/s41598-021-99015-3

[20] Vayá, M., et al. (2020). BIMCV COVID-19+: A large annotated dataset of RX and CT images from COVID-19 patients. *arXiv*. https://doi.org/10.48550/ARXIV.2006.01174

[21] Lakhani, P., et al. (2021). The 2021 SIIM-FISABIO-RSNA Machine Learning COVID-19 Challenge: Annotation and standard exam classification of COVID-19 chest radiographs. *OSF Preprints*. https://doi.org/10.31219/osf.io/532ek

[22] Rajpurkar, P., et al. (2017). CheXNet: Radiologist-level pneumonia detection on chest X-rays using deep learning. *Nature Medicine*, *25*(11), 1527-1532.

[23] Haralick, R., et al. (1973). Textural features for image classification. *IEEE Transactions on Systems, Man, and Cybernetics*, *SMC-3*, 610–621.

[24] Shu, H., et al. (2007). Moment-based approaches in imaging: Part 1, basic features. *IEEE Engineering in Medicine and Biology Magazine*, *26*(5), 70-74. https://doi.org/10.1109/emb.2007.906026

[25] Wang, X., et al. (2017). ChestX-ray8: Hospital-scale chest X-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 3462-3471).

[26] Tsai, E., et al. (2021). The RSNA international COVID-19 open radiology database (RICORD). *Radiology*, *299*(1), E204-E213. https://doi.org/10.1148/radiol.2021203957

[27] Litmanovich, D., et al. (2020). Review of chest radiograph findings of COVID-19 pneumonia and suggested reporting language. *Journal of Thoracic Imaging*, *35*(6), 354-360. https://doi.org/10.1097/RTI.0000000000000541

[28] Huang, G., et al. (2017). Densely connected convolutional networks. *arXiv*. https://doi.org/10.48550/ARXIV.1608.06993

[29] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep residual learning for image recognition. *arXiv*. https://doi.org/10.48550/arXiv.1512.03385

[30] Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. *arXiv*. https://doi.org/10.48550/arXiv.1505.04597

[31] Organización Mundial de la Salud. (n.d.). Enfermedad por coronavirus (COVID-19). Recuperado de https://www.who.int/es/emergencies/diseases/novel-coronavirus-2019

[32] National Center for Biotechnology Information. (n.d.). Coronaviruses. *NCBI Bookshelf*. Recuperado de https://www.ncbi.nlm.nih.gov/books/NBK554776/

[33] Solawetz, J., & Team Roboflow. (2020). What is YOLOv5? A guide for beginners. *Roboflow Blog*. Recuperado de https://blog.roboflow.com/yolov5-improvements-and-evaluation/

[34] Reynolds, J. H., Mcdonald, G., Alton, H., & Gordon, S. B. (2010). Pneumonia in the immunocompetent patient. *British Journal of Radiology*, 83(996), 998-1009. https://doi.org/10.1259/bjr/31200593

[35] Marikar, N. S., Khan, A., Salem, K., & Nelson, J. (2023). Pharmacological inhibition of human EZH2 can influence a regenerative β-like cell capacity with in vitro insulin release in pancreatic ductal cells. *Clinical Epigenetics*, 15(1), 101. https://doi.org/10.1186/s13148-023-01491-z

[36] Duthie, J., & Gaffney, T. A. (1984). Anatomy and physiology of respiration. *Nursing (Lond)*, 2(27), 785-787.

[37] Minchin, S., & Lodge, J. (2019). Understanding biochemistry: Structure and function of nucleic acids. *Essays in Biochemistry*, 63(4), 433-456. https://doi.org/10.1042/EBC20180038

[38] Mack, C. D., Wasserman, E. B., Killerby, M. E., et al. (2022). Effectiveness and use of reverse transcriptase polymerase chain reaction point of care testing in a large-scale COVID-19 surveillance system. *Pharmacoepidemiology and Drug Safety*, 31(5), 511-518. https://doi.org/10.1002/pds.5424

[39] Zompatori, M., et al. (1994). Diffuse ground-glass opacity of the lung. A guide to interpreting the high-resolution computed tomographic (HRCT) picture. *Radiologia Medica*, 88(5), 576-581.

[40] Murphy, K. P. (2022). *Probabilistic Machine Learning: An Introduction*. MIT Press.

[41] Mitchell, T. M. (1997). *Machine Learning*. McGraw Hill.

[42] Jocher, G., & Ultralytics Team. (2023). YOLOv5 and YOLOv8. *GitHub*. Recuperado de https://github.com/ultralytics/ultralytics

[43] Yann Henon. (2021). pytorch-retinanet. *GitHub*. Recuperado de https://github.com/yhenon/pytorch-retinanet

[44] Santos López, G., Pérez-Torres, A., Aguilar, R., & González-Pérez, M. (2021). SARS-CoV-2: Basic concepts, origin and treatment advances. *Gaceta Médica de México*, 157(1), 84-89. https://doi.org/10.24875/GMM.M21000524

[45] Pfeiffer, D., & Weisser, G. (2020). Advanced X-ray Imaging Technology. *Recent Results in Cancer Research*, 216, 3-30. https://doi.org/10.1007/978-3-030-42618-7_1

[46] Seeram, E., & Seeram, D. (2018). Computed Tomography: A Technical Review. *Radiologic Technology*, 89(3), 279CT-302CT.

[47] Larobina, M., & Murino, L. (2023). Thirty years of the DICOM standard. *Tomography*, 9(5), 1829-1838. https://doi.org/10.3390/tomography9050145

[48] Saripalle, R., et al. (2019). Using HL7 FHIR to achieve interoperability in patient health record. *Journal of Biomedical Informatics*, 94, 103188. https://doi.org/10.1016/j.jbi.2019.103188

[49] Jähne, B., et al. (2000). *Computer Vision and Applications: A Guide for Students and Practitioners*. Academic Press.

[50] Rudin, W. (1991). *Functional Analysis* (2nd ed.). McGraw-Hill.

[51] Dubey, S. R., et al. (2022). Activation Functions in Deep Learning: A Comprehensive Survey and Benchmark. *arXiv*. https://doi.org/10.48550/arXiv.2209.00722

[52] Ciampiconi, L., et al. (2023). A Survey and Taxonomy of Loss Functions in Machine Learning. *arXiv*. https://doi.org/10.48550/arXiv.2303.14145

[53] Chai, T., & Draxler, R. R. (2014). Root mean square error (RMSE) or mean absolute error (MAE)? Arguments against avoiding RMSE in the literature. *Geoscientific Model Development*, 7(3), 1247-1250. https://doi.org/10.5194/gmd-7-1247-2014

[54] Shah, A., et al. (2022). Advancement of deep learning in pneumonia/Covid-19 classification and localization: A systematic review with qualitative and quantitative analysis. *Chronic Diseases and Translational Medicine*, 8(3), 154-171. https://doi.org/10.1002/cdt3.17

[55] Mao, A., et al. (2023). Cross-Entropy Loss Functions: Theoretical Analysis and Applications. *arXiv*. https://doi.org/10.48550/arXiv.2303.14145

[56] Pearce, J. M. S., et al. (2004). Sir Charles Scott Sherrington (1857–1952) and the synapse. *Journal of Neurology, Neurosurgery & Psychiatry*, 74, 44-544. https://doi.org/10.1136/jnnp.2003.01792

[57] Leung, et al. (2009). SGD. https://doi.org/10.1007/978-3-540-29676-8_6433

[58] Liu, Y., et al. (2020). An Improved Analysis of Stochastic Gradient Descent with Momentum. *arXiv*. https://doi.org/10.48550/arXiv.2303.14145

[59] Ward, R., et al. (2021). AdaGrad stepsizes: Sharp convergence over nonconvex landscapes. *arXiv*. https://doi.org/10.48550/arXiv.2106.10019

[60] Soujanya, B., et al. (2020). Optimization with ADAM and RMSprop in Convolution neural Network (CNN): A Case study for Telugu Handwritten Characters. *International Journal of Emerging Trends in Engineering Research*, 8(9), 5759-5766. https://doi.org/10.30534/ijeter/2020/38892020

[61] Churruca, M., et al. (2021). COVID-19 pneumonia: A review of typical radiological characteristics. *World Journal of Radiology*, 13(10), 327-343. https://doi.org/10.4329/wjr.v13.i10.327

[62] Matthew, D., et al. (2012). ADADELTA: An Adaptive Learning Rate Method. *arXiv*. https://doi.org/10.48550/arXiv.1212.5701

[63] Guilhoto, et al. (2018). An overview of artificial neural networks for mathematicians. University of Chicago.

[64] Salinas-Escudero, et al. (2020). A survival analysis of COVID-19 in the Mexican population. *BMC Public Health*, 20(1616). https://doi.org/10.1186/s12889-020-09721-2

[65] Fjodor van Veen. (2016). The Neural Network Zoo. *The Asimov Institute* . Recuperado de https://www.asimovinstitute.org/neural-network-zoo/

[66] Lin, T.-Y., et al. (2018). Focal Loss for Dense Object Detection. *arXiv*. https://doi.org/10.48550/arXiv.1708.02002