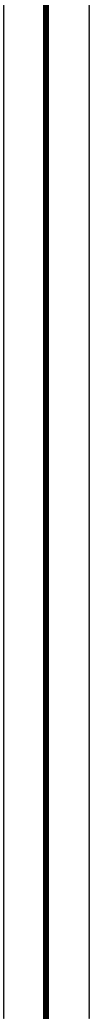
Universidad Nacional Autónoma de México



Facultad de Ciencias

Herramientas de Inteligencia artificial aplicadas al diagnóstico temprano de COVID-19 usando radiografía de tórax

T E S I S

QUE PARA OBTENER EL TÍTULO DE:

Física

PRESENTA:

Carolina Rosas Alatriste

TUTORA

Dra. Roxana Mitzayé del Castillo Vázquez

Ciudad Universitaria, CD.MX., 2022

Datos del alumno:

Rosas Alatriste Carolina

Universidad Nacional Autónoma de México, Facultad de Ciencias

Licenciatura en Física

Número de cuenta: 418003190

Asesora de tesis:

Dra. Roxana Mitzayé del Castillo Vázquez

Sinodal Propietaria:

Dra. Adriana Andraca Gómez

Sinodal Propietario:

Dr.Sergio Antonio Alcalá Corona

Sinodal Suplente:

Dr. Huziel Enoc Sauceda Félix

Sinodal Suplente:

M. en C. Edgar Vázquez Luis

DEDICATORIA

A mi madre, por siempre estar. Gracias por creer en mis sueños y ayudarme a ser quien soy hoy.

A mi padre, por ayudarme a sonreír. Gracias por los abrazos y tu complicidad.

A mis maestros, cada uno de ellos. Gracias por prestarnos su tiempo y sus conocimientos.

A mis amigos, gracias por cada risa compartida.

A mis abuelos, gracias por todo su amor.

AGRADECIMIENTOS

Agradezco el apoyo de Dirección General de Asuntos de Personal Académico (DGAPA-UNAM), con el proyecto IN120120 para la realización de esta tesis. Asímismo, agradecemos a la Dirección General de Cómputo y Tecnologías de Información y Comunicación (DGTIC), Universidad Nacional Autónoma de México (UNAM) por el apoyo de supercómputo con el proyecto LANCAD-UNAM-DGTIC-385.

Así mismo, agradezco la colaboración del Dr. Malco Enrique Flores Valenzuela, médico radiólogo del Hospital Médica Avanzada Contigo Aguascalientes por su contribución a esta tésis y por la labor que realiza todos los días.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

RESUMEN

En los últimos años se ha incrementado la implementación técnicas de inteligencia artificial para el análisis en estudios radiológicos de diferentes índoles pues, es una herramienta de diagnóstico muy eficaz tanto para médicos como técnicos radiólogos. Esto se ha implementado para la detección del SARS-CoV-2, como resultado de la neumonía que forma parte del cuadro sintomatológico de la enfermedad y, que resulta radiológicamente distinguible de neumonías virales producidas por otras enfermedades, o bien, de la opacidad pulmonar que se puede producir por diversos motivos. Una ventaja de los diagnósticos realizados mediante radiografías torácicas es el bajo costo y la disponibilidad general de este tipo de estudios.

En el presente trabajo se propone un modelo clasificador de imágenes Deep Learning con cuatro clases o categorías; paciente sano, paciente COVID, paciente con Neumonía Viral y paciente con Opacidad Pulmonar.

El modelo fue entrenado mediante una base de datos obtenida de la plataforma Kaggle y se puede consultar en [8]. La base de datos con la que se trabajó consta de 19,845 imágenes en total. Cada imagen viene etiquetada con una de las 4 clases mencionadas en el párrafo anterior. Del total de imágenes; $9,862$ corresponden con pacientes sanos, $1,015$ denotan neumonía viral, $4,252$ pertenecen a pacientes COVID y $5,682$ son de personas que por diversos motivos desarrollaron opacidad pulmonar.

Adicionalmente, se logró contactar con el Dr. Malco, médico radiólogo que labora en el hospital Médica Avanzada Contigo (MAC) en la ciudad de Aguascalientes, México. El Dr. Malco coolaboró conmigo para proporcionarme las radiografías de 64 pacientes que fueron diagnosticados con COVID- 19 mediante pruebas de PCR y que fueron internados para recibir tratamiento en el mencionado hospital. Los estudios radiológicos de los 64 pacientes fueron añadidas a la base de datos en el grupo de prueba como parte del proceso de validación cruzada. Es de destacar que la colaboración con el Dr. fue de suma importancia para el trabajo, dado que el hecho de que todas las radiografías fueron realizadas después de haberse confirmado la presencia del virus mediante la prueba PCR, nos ayuda a evaluar el verdadero desempeño del modelo.

El modelo fue probado con métricas de evaluación de clasificadores tipo límite, tras ser entrenado mediante 5 redes neuronales. Las redes neuronales elegidas fueron seleccionadas debido a características arquitectónicas de interés como el enfoque de red residual, el número de capas convolucionales,

la estructura de cuello de botella, entre otras. Es importante destacar que dado el proceso de evaluación de los resultados, nos encontramos frente a métricas que entran dentro de la descripción del método Montecarlo. Debemos recordar que el Montecarlo, es un método no determinista o estadístico numérico, usado para aproximar expresiones matemáticas complejas y costosas de evaluar con exactitud. Por lo que los porcentajes presentados en la presente tésis son las aproximaciones a un resultado máximo a obtener en el caso de un experimento en particular.

Habiendo mencionado lo anterior, la red neuronal con la que el modelo tuvo resultados destacables fue ResNet50, alcanzando 95 % de exactitud global y exactitud mínima por clase de 89 %. El modelo mostró tener una buena evolución en el entrenamiento con un buen ajuste y generalización al realizarse pruebas con grupos de validación cruzada tras entrenarse con dicha red.Asimismo, en las etapas de prueba, se alcanzaron rangos de error máximo de 11 %, además de precisiones sensibilidades y especificidades por categoría de al menos 79.3 % y hasta el 100 %.

Los resultados encontrados con la red ResNet50 nos dan indicios de la viabilidad de usar el modelo propuesto como una herramienta diagnóstica para un especialista. Sin embargo, es importante recordarle al lector las condiciones de evaluación de los métodos descritos en este trabajo, por lo que resultaría importante realizar pruebas más sofisticadas que evalúen las probabilidades individuales de cada predicción como pueden ser las curvas ROC o los mapas de calor. El trabajo realizado en la presente tesis es un

buen precedente para un proyecto con evaluaciones más estrictas y que, por decir, se pudiera usar con tranquilidad como herramienta en el ámbito médico.

Ahora bien, en lo que refiere a las demás redes, en todos los casos se obtuvieron precisiones, sensibilidades y especificidades mayores al 85 % para las 4 categorías. Además, en todas las redes se obtuvieron rangos de error menores al 1 %.

Particularmente, en el caso de la Neumonía Viral, encontramos que sus evaluaciones se encuentran sobre el 92 % en todos los casos. Encontramos pues, que el modelo es capaz de identificar correctamente tanto los casos positivos verdaderos como los negativos verdaderos en lo que corresponde a esta categoría. Esto es importante porque, también hemos visto que, de acuerdo con H. S. Maghdid y colaboradores [9], una de las principales limitantes del diagnóstico clínico de pacientes COVID mediante estudios radiológicos es la similitud que tiene en el sentido radiológico con la Neumonía Viral.

Del mismo modo, en el caso de la opacidad pulmonar, encontramos resultados mayores al 82 % en las tres métricas, además de un rango de error del 7 %. Aunque no alcanzan los porcentajes de otras categorías, aún se encuentran en un rango aceptable.

Por lo que podemos concluir que se cumplieron los objetivos de la tesis tras lograr la construcción de un modelo que, mediante el entrenamiento

con redes neuronales pre entrenadas, permite la detección de las 4 clasificaciones. Además se determinó que el modelo trabaja mejor tras ser entrenado con la red ResNet50 y los resultados encontrados, aunque no nos hablan por completo del desempeño del modelo, alientan a continuar la línea de investigación.

CAPÍTULO 1

INTRODUCCIÓN

Del artículo de Varshni, D. et. al. [10] y Chaturvedi, S.S. et. al. [11] sabemos que en los últimos años se ha incrementado la implementación de técnicas de tipo *deep learning* para el análisis en estudios radiológicos de distinto tipo de enfermedades presentes en la cavidad torácica, pues permite aligerar y optimizar la labor de los médicos y técnicos radiólogos. Esto se ha implementado para la detección del SARS-CoV-2 y otras enfermedades. De hecho, de acuerdo con el grupo de Zhang et. al. [12], se ha demostrado que es posible detectar el SARS-CoV-2 o COVID-19 gracias a la neumonía que se produce como parte del cuadro sintomatológico de la enfermedad y, que resulta radiológicamente distinguible de neumonías virales producidas

por otras enfermedades, o bien, de la opacidad pulmonar que se puede producir por diversos motivos.

La neumonía, en términos generales, es una infección que produce inflamación alveolar ya sea en uno o ambos pulmones del paciente infectado. Existen muchas causas para la neumonía, dentro de las cuales se encuentra la producida por bacterias, hongos o bien, por un virus como el del COVID-19. Casi el 30 % de los casos de neumonía son virales y suelen sanar por sí mismos, sin embargo, la neumonía causada por el COVID-19 puede volverse muy agresiva causando la muerte [9],[13].

La prueba estándar para el diagnóstico del virus SARS-CoV-2 es la *"Viral nucleic acid detection using real-time polymerase chain reaction"* RT-PCR por sus siglas en inglés. Sin embargo, de acuerdo con el trabajo de Watson y sus colaboradores [14], se ha demostrado que la prueba no es del todo óptima pues presenta una sensibilidad del 70 % y una especificidad del 95 %. Por lo cual, si un grupo de 100 personas se realizara la prueba, encontraríamos que 24 personas tendrían un falso negativo, sin mencionar que en el caso de una mutación la prueba RT-PCR se vería obsoleta hasta que se realice la correspondiente secuenciación del ADN. Todo lo anteriormente mencionado nos motiva a buscar métodos de diagnóstico complementarios a la prueba RT-PCR.

Ahora bien, la radiografía de Tórax (RxT) es una modalidad de imagen presente en la mayoría de los puntos de atención médica-hospitalaria y,

en general, es el primer paso cuando se tiene sospecha de neumonía. Es más, en la infección producida por el SARS-COV-2 confirmada por laboratorio con cuadro clínico moderado a severo, la evaluación radiográfica estará encaminada a brindar una escala del grado de afectación pulmonar inicial y para la evolución en la hospitalización. Así mismo, en el caso de sospecha aún sin resultados de laboratorio, la evaluación radiográfica puede establecer el grado de afección. Baldomá y su grupo de investigación [15] han comprobado que la radiografía de Tórax se debe mantener como método útil de cribado en las etapas medias de la enfermedad, cuando la RxT resulta más sensible para detectar la afección pulmonar.

Por otra parte, el diagnóstico de la neumonía producida por el virus Sars-CoV2 mediante radiografías torácicas representa una gran ventaja para combatir la pandemia en la cual nos encontramos, pues a diferencia de la prueba RT-PCR, es mucho más accesible y económica. De acuerdo con el estudio del grupo de Wong [16], la sensibilidad del diagnóstico a ojos de médicos radiólogos mediante el uso del mencionado estudio es tan sólo del 69 % comparado con el 91 % que representa la RT-PCR. Sin embargo, en años recientes, los modelos *Computer Aided Designs* CAD por sus siglas en inglés, han demostrado facilitar la labor en el área médica, principalmente en la detección de Cáncer de Mama, nódulos pulmonares, mamogramas, entre otros. Particularmente, en el campo de detección del COVID-19 y Neumonía Viral mediante modelos CAD, se han encontrado resultados de diagnóstico con sensibilidades de hasta 83.61 %, 96.8 % y 80.02 % en el 2020

[12], [17], [18] y [9], resultados que son comparables con el desempeño de un especialista e incluso con los resultados obtenidos por la prueba RT-PCR.

El Sars-CoV2 es un virus esférico, con un diámetro aproximado de 60-140 nm, y envuelto en proteínas espiculares o "Spikes" de 8 nm a 12 nm de longitud. Su estructura está conformada por dos partes, una nucleocápside fosforilada que tiene la función de proteger el material genético y se encuentra dentro de la bicapa de fosfolípidos de la envoltura. La segunda parte es una envoltura externa en donde se encuentran las proteínas estructurales que cumplen funciones de anclaje y propagación [19], [14]. Por lo que es un virus particularmente propenso a la mutación. Las pruebas de PCR se basan en dos reacciones consecutivas: la primera se basa en la conversión de RNA en DNA complementario (DNAC) a través de la enzima de transcriptasa inversa; y la segunda es la amplificación de la muestra de DNAC por reacción en cadena de la polimerasa utilizando cebadores específicos de genes y sondas de hidrólisis marcadas con fluorescencia [13]. Por lo que es necesario detectar la variante y realizar el proceso antes mencionado para cada una de las variantes individualmente. La RT-PCR actualmente es el estándar dorado de las pruebas para detección del virus del COVID-19, sin embargo, una herramienta radiológica puede ayudar a detectar pacientes con el virus aún si no se ha tipificado.

De este modo, considerando las especificaciones realizadas anteriormente en pro del desarrollo de herramientas computacionales que sirvan para

la detección de diversas patologías pulmonares, como es el caso de la neumonía producida como parte del cuadro de sintomatología del COVID-19. En el presente trabajo se busca crear un modelo de inteligencia artificial (IA) funcional que pueda usarse como herramienta diagnóstico o didáctica para la detección radiológica de la enfermedad producida por el SARS-COV-2 y sus variantes. Particularmente, se busca la creación de un modelo de deep learning que se apoye en redes neuronales convolucionales para su entrenamiento. Dicho modelo será evaluado tras ser entrenado mediante 5 redes neuronales de vanguardia y se determinará con cuál red se encuentra un mejor desempeño.

CAPÍTULO 2

MARCO TEÓRICO

2.0.1. Redes Neuronales

Las neuronas piramidales o células pirámides son un tipo de neuronas multipolares situadas en diversas partes del cerebro, principalmente en la corteza cerebral. De hecho, las neuronas piramidales conforman la mayor parte de la corteza y se encuentran en hasta 6 capas de esta. La particularidad que hace que nos fijemos en las células pirámide es la forma en la que se comunican entre sí, encontrando pequeñas líneas de salto entre las capas. De manera análoga, las redes neuronales pueden ensamblar construcciones de aprendizaje usando atajos sobre las capas. Los saltos contienen norma-

lización por lotes sin linealidad entre ellos. Y mediante una matriz de peso extra, podemos aprender los "pesos de salto" en algunos casos, como lo son las redes de autopista.

El sistema nervioso de los animales se estructura con neuronas que forman redes para el procesamiento de la información generada mediante estímulos generados por los sistemas sensoriales. Las redes neuronales artificiales (RNA) surgen a partir de las ideas expresadas por Mc Culloch y Pitts en su artículo [20]. En este artículo se postula que las neuronas funcionan como variables booleanas. Este postulado fue criticado como teoría biológica, pero permitió generar una neurona como un modelo lineal seguido de una función activación booleana. Aquí, la función lineal representa la sinapsis (unión entre neuronas) y la agregación de la información, mientras que la función no lineal representa el procesamiento que hace la neurona. Esto resultó en un modelo como el de la figura 2.1. La ecuación es:

$$r = \sum_{i=1}^{n} x_i w_i + b, \tag{2.1}$$

donde x_i y w_i representa los datos de entrada y los pesos sinápticos respectivamente, además, b es un factor de polarización. Tenemos que el resultado para r es procesado por una función binaria que arroja un uno o cero respecto al valor de r.

El perceptrón fue el primer acercamiento a una red neuronal. El perceptrón utiliza una neurona artificial como la de la figura 2.1. Tenía su arquitectura en tres capas y un algoritmo de aprendizaje. Este fue proba-

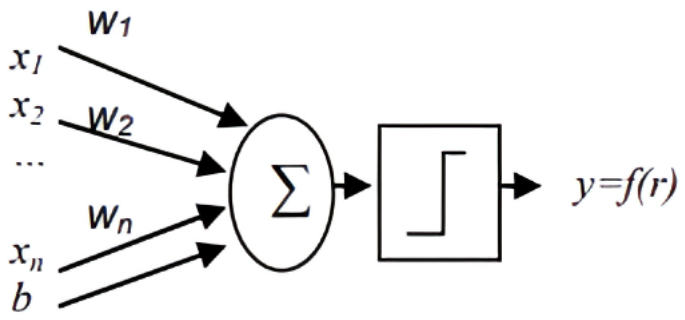


Figura 2.1: Modelo de neurona artificial con salida binaria. Figura obtenida de [1]

do como un detector de caracteres ópticos, por lo que el campo de redes neuronales se inicio como una forma de procesar imágenes. Luego del perceptrón y otras redes con neuronas binarias, surgen otros modelos que también han sido utilizadas en el área de procesamiento de imágenes [21]. Un ejemplo importante son las redes con funciones de activación continuas, tales como las sigmoidales o las funciones de base radial, definidas en las ecuaciones de abajo [22], [23]

$$f(r) = \frac{1}{1 + \exp(-\alpha r)} \tag{2.2}$$

$$f(c, x) = |x - c| \tag{2.3}$$

El parámetro α caracteriza la inclinación de la función, c es un punto llamado centro. Podemos decir que, en general, los modelos clásicos de estas redes tienen la característica de ser supervisados, es decir, necesitan una referencia en un proceso de entrenamiento para obtener una salida deseada a una entrada determinada. También se han generado modelos con características no supervisadas como los mapas auto-organizados de Kohonen (SOM) y la Teoría de Resonancia Adaptiva (ART). Además han surgido

otros modelos como las Redes Neuronales Recurrentes, y las que se basan en modelos probabilísticos que han sido muy útiles en el procesamiento de imágenes por varios años. En los últimos 10 años, han sido varias las arquitecturas de redes neuronales utilizadas para el procesamiento de imágenes. Algunos ejemplos de dichas redes incluyen las redes neuronales recurrentes (RNN), perceptrón multicapa (MLP), red neuronal pulsocortada (PCNN), redes neuronales basadas en modelos probabilísticos (RPNN), entre otras. En este trabajo nos vamos a enfocar particularmente a las redes neuronales convolucionales (CNN)[23],[24],[3].

De hecho, en su libro, Chollet, F. [3] indica que una característica es una transformación de los datos que se diseña para que sea más fácil de modelar. Sabemos que en el contexto de una imagen, una característica es un atributo visualmente distintivo. Por ejemplo, podemos distinguir un perro de un conejo por la forma de sus orejas. De este modo, nuestro objetivo es extraer información sobre los bordes de una imagen determinada y luego usar esa información como nuestras características, en lugar de los simples píxeles en bruto. Para hacerlo, usamos algo llamado convolución. Una convolución no requiere nada más que la multiplicación y la suma: dos operaciones que son responsables de la gran mayoría del trabajo que veremos en modelos de aprendizaje profundo [25].

Ahora, tal como explica Albon, C. en su libro [26], en el campo de la teoría de la señal, la convolución ocurre entre dos señales continuas unidimensionales $f(t)$ y $g(t)$. Para comprender el significado de la operación

de convolución entre señales es necesario introducir el concepto de sistema lineal de tiempo invariante (LTI). La teoría de la señal ofrece las herramientas para analizar el comportamiento de los sistemas físicos observando su comportamiento en respuesta a ciertos estímulos de entrada. De este modo, siendo S el sistema a modelar. S acepta un estímulo en la entrada y produce una señal de salida. Diremos que S es un sistema LTI si se verifican las siguientes propiedades [22], [27],[26]:

1. Linealidad: $S[\alpha x_1(t) + \beta x_2(t)] = \alpha y(x_1(t)) + \beta y(x_2(t))$

2. Tiempo de invarianza: $S[x(t + t_0)] = y(t + t_0)$

Donde α y β son números reales. Se puede analizar el comportamiento de un sistema LTI analizando su respuesta a la función Delta de Dirac $\delta(t)$ [1]. Notemos que si establecemos $\delta(t)$ en la entrada del sistema S, obtendremos la respuesta del sistema con un impulso unitario centrado en el cero.

Ahora bien, definiremos la respuesta impulsiva de un sistema a la salida del sistema cuando la función Delta de Dirac está presente en la entrada

[1] $\delta(t)$ es una función que vale cero en cualquier punto de su dominio, excepto por el punto cero, ahí, su valor es un infinito de un grado lo suficientemente alto como para hacer realidad la función [28]

$$\int_{-\infty}^{\infty} \delta(t)\phi(t)\ dt = \phi(0)$$

(2.4)

como

$$h(t) = S[\delta(t)] \tag{2.5}$$

La respuesta impulsiva tiene mucha importancia ya que nos permite calcular la respuesta del sistema LTI en cualquier entrada. Una señal genérica $x(t)$ se puede expresar mediante la aplicación $\delta(t)$ trasladada en cualquier punto del dominio de $x(t)$. Tenemos que la convolución entre dos señales es [22], [26],[25]

$$x(t) = \int_{\infty}^{\infty} x(t)\delta(r-\tau)\ d\tau = x(t) * \delta(t) \tag{2.6}$$

donde $*$ denota la convolución. De la ecuación (2.6) podemos resaltar la propiedad recursiva que yace en la definición de la convolución. Además, podemos notar que efectivamente, $\delta(t)$ es un elemento neutro, de modo que es posible analizar la convolución para comprender el motivo por el que es la operación fundamental en el análisis de las señales. Si $x(t)$ es una señal genérica de entrada y $h(t)$ es la respuesta impulsiva del sistema S, entonces el resultado de la convolución, $y(t)$, se define como

$$y(t) = x(t) * \delta(t) = (x * h)(t) = \int_{-\infty}^{\infty} x(t)\delta(t-\tau)\ d\tau \tag{2.7}$$

y representa el comportamiento del sistema LTI modelado por su respuesta impulsiva h(t) cuando x(t) se coloca en su entrada. Este resultado es de fundamental importancia ya que muestra como la respuesta impulsiva caracteriza totalmente el sistema.

2.0.2. Convolución unidimensional

La convolución unidimensional discreta también se puede definir. De acuerdo con Huang, J. & Ling, C.X. en [29], si $g(n)$ y $x(n)$ son definidos en el campo de los números enteros $\mathbb{Z}$, entonces es posible calcular su convolución mediante:

$$(x * g)[n] = \sum_{m=-\infty}^{\infty} x[m]g[n-m] \tag{2.8}$$

La generalización al caso bidimensional es natural, ya que en el campo de la visión artificial la convolución se lleva a cabo principalmente entre señales discretas bidimensionales, por lo que se ocupa la parte discreta de la Delta de Dirac, es decir, la Delta de Kronecker [2]. En el caso bidimensional, el tratamiento para los sistemas LTI es generalizado y hablamos de sistemas lineales de espacio invariante (LSI).

Se define filtro o núcleo o matriz de convolución a la respuesta de un sistema LSI discreto. El filtro tiene un tamaño de $2k \times 2k$, donde k es un valor arbitrario que define la cantidad de valores de respuesta impulsiva que hay en la muestra. Así mismo, según Petersen, E. et. al. en [30], si consideramos la altura, h y el ancho, w, $h[n]$ es un filtro de dimensiones $2k \times 2k$ e I es una imagen con escala de grises, cada punto de coordenadas

[2] La delta de Kronecker puede expresarse independientemente de la dimensión del espacio que se utiliza:

$$\delta_{i,j} = \begin{cases} 1 \text{ si i=j} \\ 0 \text{ en otro caso} \end{cases} \tag{2.9}$$

(i, j) con señal resultante de la convolución entre h e I está dada por

$$O(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} h(u,v) I(i-u, j-v) \tag{2.10}$$

Intuitivamente, una convolución bidimensional consiste en filtrar una imagen de dimensiones $(2k+1) \times (2k+1)$ en la imagen I y calcular la operación que se ha indicado en (2.10) para cada píxel centrado en dicha imagen. Y recordando que si consideramos la altura, h y el ancho, w, se pueden definir algunos parámetros para la convolución en cada dirección como:

- Paso ($S_{w,h}$): Es la distancia expresada en píxeles que transcurre entre aplicaciones sucesivas, en la dirección dada, en el paso elemental de la convolución. Intuitivamente es la distancia entre el centro de la imagen que se filtra cuando ésta pasa de la posición (i, j) a la posición $I_{i+Sw,j}$ (o $I_{i,j+Sh}$ dependiendo de la dirección en la que se mueva la imagen).
- Relleno cero ($P_{w,h}$). Número de ceros para añadir como borde al resultado de la convolución.

Así, asumiendo que $S = S_h = S_w$, $P = P_h$, $I = I_h = I_w$, la imagen filtrada tendrá dimensiones de

$$O_w = O_h = \frac{I_s - k + 2P}{S} + 1. \tag{2.11}$$

Que tendrá como resultado un valor entero, se debe elegir correctamente el valor de peso tal que sea compatible con el tamaño de la imagen de entrada. A lo largo de los años se acostumbraba que, la parte de la visión

artificial que se ocupa del procesamiento de imágenes consiste en definir filtros manualmente con el objetivo de mejorar la calidad de las imágenes o para extraer sus características [31], [30]. Entre los ejemplos más conocidos encontramos el filtro Gaussiano para la reducción del ruido y el filtro Sobel para la extracción de las derivadas direccionales de la imagen. En la figura 2.2 se muestra el resultado de la convolución de una imagen con núcleo de detección de bordes. La extracción de características de una imagen es

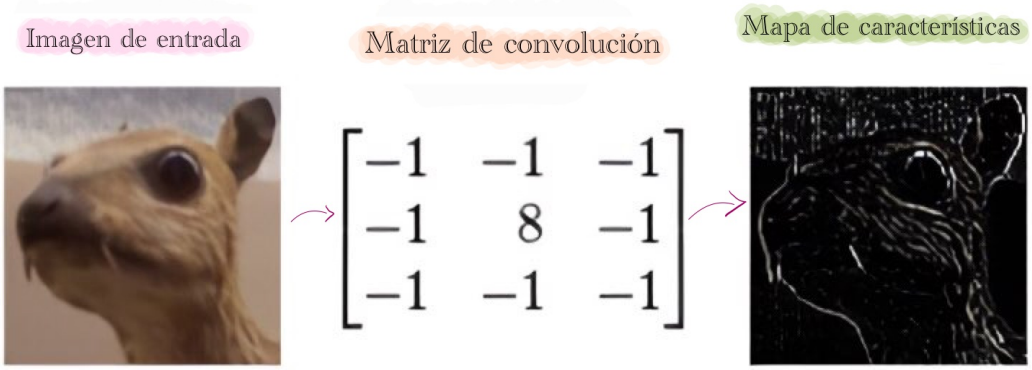


Figura 2.2: Extracción del borde de una imagen mediante una convolución. Figura traducida y modificada obtenida de [2]

un paso fundamental para poder crear modelos de clasificación. De hecho, después de extraer ciertas características es posible analizar el contenido de la imagen. Antes de profundizar en los filtros de convolución, es necesario extender la definición de convolución dada por la ecuación (2.10) a imágenes con más de una dimensión.

2.0.3. Convolución entre volúmenes

En general, podemos identificar un volúmen como una estructura geométrica con ancho $W \geq 1$, una altura $H \geq 1$ y una profundidad $D \geq 1$. De este modo, el caso bidimensional es una imagen en escala de grises que tiene una profundidad $D = 1$. Contrariamente, una imagen con tres canales de colores (RGB) tiene una profundidad $D = 3$. En la ecuación (2.10), se ha definido la convolución para una profundidad $D = 1$. Notemos que se puede generalizar la definición de convolución para una D arbitraria aplicando (2.10) para cada valor de D y sumando los resultados para obtener un único valor como resultado de la convolución [30]. De modo que el filtro ahora es un volúmen con profundidad D, por lo que se obtiene un filtro bidimensional para cada nivel del volumen de entrada. Es decir, para $I = (W_I, H_I, D_I) = I_1, .., I_D$ el volumen de entrada, sea $F = (2_k, 2_k, D_I) = F_1, .., F_{DI}$ el volumen del filtro, que pertenece al conjunto de volúmenes de los filtros es $(2_k, 2_k, D_I)^{F_d}$. Ahora el resultado de la convolución entre el volumen I y F pertenece a $(2_k, 2_k, D_I)^{F_d}$ tal que, el n-ésimo volumen del filtro viene dado por

$$O_n(i,j) = \sum_{d=1}^{D_I} \sum_{u=-k}^{k} \sum_{v=-k}^{k} F_d(u,v) I_d(i-u, j-v) \tag{2.12}$$

El valor de los volúmenes de los filtros depende de los valores de entrada en el entorno bidimensional del punto en el que estamos calculando la convolución a lo largo de toda la profundidad del volumen de entrada. Tenemos que el resultado de la convolución para todos los volúmenes del

filtro, es un volumen de profundidad F_d dado por la unión de los resultados convolucionales de la ecuación anterior, obtenemos

$$O = \bigcup_{i=1}^{F_d} O_i \tag{2.13}$$

Como en el caso unidimensional, el mapa de características resultante habrá dado dimensiones dependiendo del relleno cero P y del paso S elegidos. La convolución captura información espacial dada por las dimensiones del filtro de convolución combinándolo con la información presente en cada nivel, en la misma posición espacial y del volumen de entrada [29].

2.0.4. Aprendizaje en filtros de convolución

De (2.12) sabemos que la convolución de un volumen con profundidad D requiere D filtros para producir un volumen con $D = 1$. Entonces la convolución extrae sólo una característica del volumen de entrada. Esta característica puede o no ser suficiente para ser utilizada como único vector de características por un algoritmo de Machine Learning.

Las características extraídas de la operación de convolución tienen una disposición espacial, y por lo tanto, no son vectores de características, sino mapas de características. Es posible obtener un vector de características de un mapa de características linealizando el mapa $O \times O$ en un vector O^2.

Ahora, de acuerdo con Jeong en [2], si estamos interesados en buscar características básicas y elementales, como las líneas horizontales o verticales,

o un color en particular, entonces una sola característica puede ser suficiente. Sin embargo, si estamos interesados en la investigación de un patrón más complejo, como en el caso de este trabajo, una sola característica no es suficiente para caracterizar la complejidad y la variabilidad (posiciones, dimensiones o colores) de la imagen.

Definir manualmente los filtros capaces de extraer características es un proceso largo y complejo que no siempre conduce a los resultados deseados. Por esta razón, se puede usar Machine Learning para aprender los valores de los filtros, para que la combinación de las características extraídas de los filtros pueda caracterizar completamente a la imagen que está en estudio. Para hacer esto, es posible utilizar las redes de neuronas artificiales. Al colocar las neuronas en cuadrículas de dimensiones $2k \times 2k$, pueden usarse como filtros de convolución para la extracción de las características y modificar sus valores (y por lo tanto, las características extraídas) durante los pasos de la propagación hacia atrás.[25], [2]

El proceso de entrenamiento mostrará las imágenes a la red de los procesos a clasificar y el valor de la etiqueta real y la pérdida también requiere el valor de la predicción. Por este motivo es necesario que se agreguen capas de clasificación totalmente conectadas a la red para realizar la combinación de las características extraídas y obtener el valor de la clase y.

De acuerdo con Chollet en su libro [3], dado que es muy difícil delimitar la cantidad de características necesarias para caracterizar por completo las

entradas, podemos definir un hiperparámetro F_d o profundidad de filtro, que hace referencia a la cantidad de filtros de convolución que se intentan aprender. Cada uno de estos filtros está compuesto de D filtros $2k \times 2k$. Por lo tanto, una convolución que acepta un volumen de entrada $(W; H; D)$ se lleva a cabo con F_d filtros, cada uno compuesto por D filtros, produce un volumen a su vez con profundidad F_d.[32]

El conjunto de los F_d filtros se denomina capa de convolución [28]. Durante el paso hacia adelante, el volumen de la entrada tiene un filtro extraído de la capa de convolución, con igual profundidad a la profundidad del volumen de entrada. El resultado de la convolución con el filtro genera un mapa de características. El conjunto de mapas de características, generado por F_d filtros genera un volumen con profundidad F_c que a su vez puede ser utilizado para extraer otras características. Como es fácil de intuir, el número de parámetros aumenta a mayor número de capas de convolución. Un método para reducir el número de parámetros sin reducir la efectividad del aprendizaje es el pooling, pero antes de definir lo que es una capa de agrupación o pooling layer debemos ver los componentes básicos de una red CNN.

2.0.5. Componentes en las arquitecturas de las CNN

Capa convolucional (Convolutional layer)

Es la capa principal de las redes de neuronas convolucionales, siendo indispensable el uso de una o de más capas de este tipo en dichas redes. Los parámetros en una capa convolucional, en la práctica, se refiere a un conjunto de filtros entrenable. Cada filtro ocupa un espacio pequeño a lo largo de las dimensiones de anchura y altura, pero se extiende por toda la profundidad del volumen de entrada al que se aplica. Durante la propagación directa o hacia delante, cada filtro a lo largo de la anchura y de la altura del volumen de entrada, produce un mapa de activación bidimensional (o mapa de características) para ese filtro [3]. A medida que el filtro se mueve a lo largo del área de la entrada, se efectúa un producto escalar entre los valores del filtro y los de la región de entrada a la que se aplica. Poner en cola todos estos mapas de características para todos los filtros, a lo largo de la dimensión de la profundidad forma el volumen de salida de una capa convolucional. Y como sabemos de Howard en [25], cada elemento de este volumen puede ser interpretado como la salida de una neurona que solamente se observa en una pequeña región de entrada y que comparte sus parámetros con otras neuronas en el mismo mapa de características, ya que todos estos valores provienen de la aplicación de un mismo filtro.

Como ya he mencionado en secciones anteriores, una característica en el contexto de una imagen es un atributo visualmente distintivo, por ejemplo,

podemos distinguir un perro de un oso por sus orejas, nariz, dimensiones, etc. Las convoluciones nos permiten extraer información de las imágenes y usar esa información como características. En otras palabras, la convolución aplica un kernel a través de una imagen, tal como se muestra en la figura 2.3. La cuadrícula de 7×7 representa la imagen sobre la cual se está aplicando el kernel. La operación convolución multiplica cada elemento del kernel por cada elemento de un bloque 3×3 obtenido de la imagen. Los resultados de esta multiplicación son sumados. En el desarrollo de este

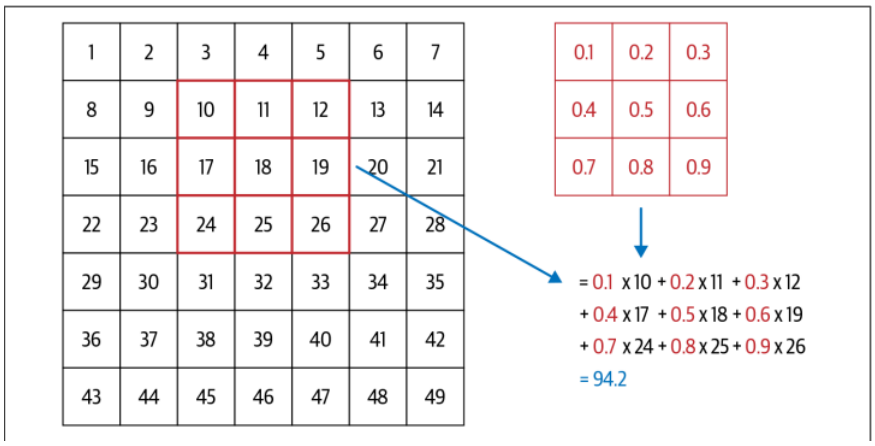


Figura 2.3: Aplicación de un kernel a una locación de una imagen. Figura tomada de [3]

trabajo se utilizó la librería PyThorch. Y tal como nos explica Chollet en su libro [3], PyThorch ya tiene implementada la operación convolución. La función se llama mediante el comando `F.conv2d` y para poder utilizar la función, es necesario agregar los siguientes parámetros:

- `input`: tensor de entrada con la forma $(minibatch, in\_channels, iH, iW)$
- `Weight`: filtros de la forma $(out\_channels, in\_channels, kH, kW)$

donde iH, iW hacen referencia a la altura y el peso de la imagen respectivamente e igualmente, kH y kW son la altura y el peso del kernel 3×3. A pesar de la aparente incongruencia entre el tamaño de los tensores, una de las mayores ventajas a la hora de usar PyThorch es que permite aplicar la convolución a varias imágenes a la vez, además de aplicar varios kernels pequeños simultáneamente.

Capa de agrupación o pooling layer

Entre sucesivas capas convolucionales, es usual colocar lo que se denomina como capa de agrupación en el medio de dichas capas. De forma concisa, la capa de agrupación toma los mapas de características producidos en la capa de convolución y los agrupa en una imagen. En esta capa, se produce la reducción de dimensionalidad reduciendo de esta forma, la complejidad que posee el modelo ayudando a evitar el sobreajuste del mismo. Es decir, lo que hacen las capas de agrupación es simplificar la información en la salida de la capa convolucional. Tal como nos explica Mattich en su artículo [1], el pooling es una operación que permite reducir la cantidad de parámetros de la red, por lo general, la operación describe la agrupación de manera operacional. Y como hemos visto, es posible imaginar la operación de agrupación con la de convolución entre volúmenes. Dado un volumen $I = (W_I; H_I; D)$ la operación de agrupación produce un volumen de salida $O = (W_O; H_O; D)$ deslizando un filtro $2k \times 2k$ que no se aprende, sino que solamente se utiliza para definir el cuerpo receptivo de la operación [28].

Capa de unidades lineales rectificadas ReLu (Rectified Units Layer)

De acuerdo con Nielsen [22], ReLu es la abreviatura de Rectified Linear Units, es decir, unidades lineales rectificadas. Este tipo de capas es muy común en una red de neuronas convolucional y se puede utilizar múltiples veces en una misma red de neuronas artificiales, frecuentemente después de cada capa convolucional. Y tal como se nos explica ampliamente en el artículo de Szgedy, C. et. al. [33], la función de esta capa es la de aumentar la propiedad de no linealidad de la función de activación, sin que se modifiquen los cuerpos receptivos de la capa convolucional. Una función muy utilizada en las capas ReLu es $f(x) = max(0, x)$, aunque también es posible otras funciones como $f(x) = tanh(x)$, $f(x) = |tanh(x)|$ o la función sigmoide $f(x) = \frac{1}{1+e^{-x}}$. Además, es posible utilizar en una capa una función cualquiera y en otra de ellas otra diferente, tal como se ejemplifica en redes como la propuesta por Xie, S. y sus colaboradores en [5]. Así mismo, en [34], Krizhevsky, A. y colaboradores explican que la función $f(x) = tanh(x)$, permite entrenar con una mayor rapidez y con un rendimiento similar a las capas ReLu. En este tipo de capas no existe ningún parámetro a establecer, simplemente se utiliza una función fijada. En su libro [35], Marsland, S. expone como estas capas tampoco tienen parámetros entrenables, por lo que tienen una propagación hacia atrás simple: se propagan hacia atrás los errores calculados hasta ese momento que provienen de la capa posterior, pasándolos a la capa anteriormente considerada.

Capa completamente conectada (fully connected layer)

Este tipo de capa es exactamente igual a cualquier capa de una red de neuronas artificial clásica, pero tiene su arquitectura totalmente conectada. Todas las neuronas de esta capa se encuentran conectadas a todas las neuronas de la capa anterior y más específicamente, se encuentran conectadas a todos los mapas de características de la capa anterior.

A diferencia de lo que se ha visto hasta ahora en las redes de neuronas convolucionales, y como nos explica Jeong, J. en [2] no se utiliza la propiedad de conectividad local. Una capa completamente conectada se encuentra conectada a todo volumen de entrada, por lo que tiene un gran número de conexiones, mientras que las capas convolucionales se encuentran conectadas a una sola región local en la entrada y que muchas de las neuronas de la capa convolucional comparten parámetros.

Además, de acuerdo con Marsland, S. en [35], la función principal de las capas completamente conectadas es llevar a cabo una especie de agrupación de la información que se ha obtenido hasta ese momento, que servirá en cálculos posteriores para la clasificación final. En general, en las redes de neuronas convolucionales, se suele utilizar más que una capa completamente conectada en serie y la última de ellas tendrá el parámetro K, que es el número de clases que se encuentran presentes en el conjunto de datos. En estas capas se conectan sus K neuronas con todos los volúmenes de entrada de cada una de sus K neuronas. Su salida será un único vector

$(1 \times 1 \times K)$, que contiene las activaciones calculadas. Este hecho, de pasar de un volumen de entrada en 3 dimensiones hasta un solo vector de salida con una sola dimensión, sugiere que después de esta capa, no haya ninguna capa de convolución. Los valores finales de K serán alimentados a la capa de salida, que a través de una cierta función probabilística realizará la clasificación tal como explica Howard, J. en [25]. Hay más parámetros que se pueden configurar, como los valores de los pesos y el sesgo, pero no es estrictamente necesario y se pueden utilizar valores predeterminados.

Capa de pérdida (Loss layer)

En esta capa es donde se comparan las predicciones con los valores reales de las imágenes. Cuando se trata de clasificar y elegir entre K posibles niveles, se usaría un clasificador de pérdidas softmax. El uso de una función euclídea también es usual con el propósito de regresión contra las etiquetas de las imágenes. Sus funciones están dadas por:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \tag{2.14}$$

$$z_j = \sum_{k=0}^{d} W_{ik} x_k \tag{2.15}$$

donde z_j es un vector de probabilidades a posteriori, j representa la i-ésima neurona de la capa de salida, o la capa de pérdida. K representa el número total de la neuronas de la capa de pérdida. W_{ik} son los pesos, x_i son los valores de entrada que recibe la capa de pérdida, $\sigma(z)_j$ es la activación de las K neuronas de la capa de pérdida.[35, 9]

Hiperparámetros

Los conceptos definidos hasta ahora están delimitados en uno o más grados de libertad por sus hiperparámetros. En la tabla 2.1 se muestran todos los hiperparámetros relacionados con la operación de convolución entre volúmenes. Notemos que la elección de algoritmo de optimización también es un hiperparámetro.

Tabla 2.1: Hiperparámetros de la convolución entre volúmenes.

F_d	Número de filtros que forman la red convolucional
$2k \times 2k$	Área del filtro de convolución/pooling
S	Paso de la operación de convolución/pooling
P	Número de ceros de relleno para la salida de la convolución/pooling

Además de los hiperparámetros que se muestran en la tabla 3.1, cada arquitectura tiene muchos otros grados de libertad. Algunos de ellos incluyen las dimensiones de entrada, el número de capas de convolución, la posición de la operación de agrupación, el tamaño de filtro, la no linealidad o la topología de la red.

2.0.6. VGG16

Conforme pasa el tiempo, las redes neuronales tienden a ser más profundas, esto se debe a que la forma más sencilla de mejorar el rendimiento de las redes neuronales profundas es mediante el aumento del tamaño [36]. La

gente de Visual Geometry Group (VGG) inventó el VGG-16 que contiene 13 capas convolucionales y 3 fully connected, llevando consigo la tradición ReLU que se implementó con la AlexNet [3]. VGG-16 apila pues, más capas que AlexNet y además usa filtros de menor tamaño (2x2 y 3x3). Consta de 138M de parámetros y ocupa aproximadamente 500 MB de espacio de almacenamiento. También se diseñó una variante más profunda, la VGG-19, que tiene 16 capas convolucionales y 3 Fully connected.

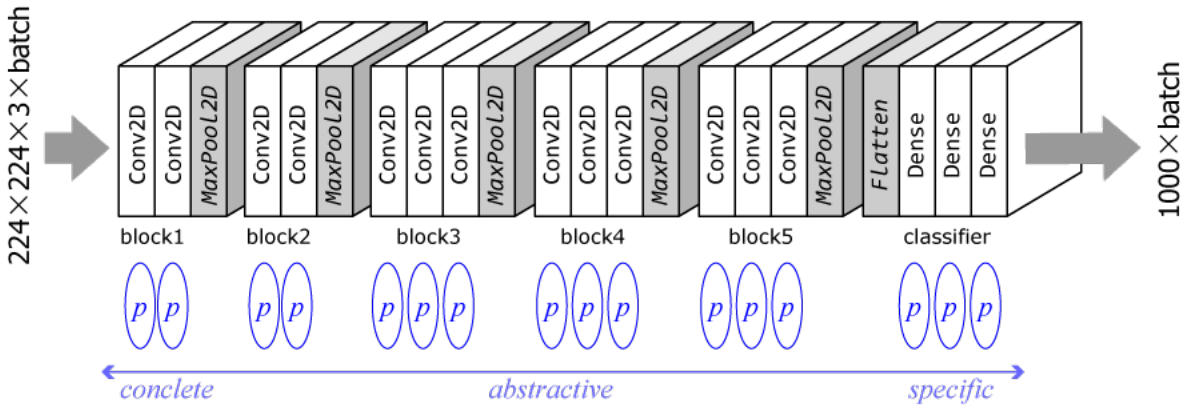


Figura 2.4: Arquitectura de la red neuronal VGG-16. Figura obtenida de [4]

2.0.7. Inception

Esta red es una colaboración entre Google, la Universidad de Michigan y la Universidad de Carolina del Norte. La construcción de esta red es de forma recursiva, es decir, mediante módulos, en lugar de apilar capas convolucionales. De ahí el nombre de Inception (tal como en la película *In-*

[3]La red AlexNet cuenta con 8 capas: 5 convolucionales y 3 fully connected. En su momento de publicación, 2012, los autores [34] señalaron que su arquitectura era una de las redes neuronales convolucionales más grandes hasta la fecha en los subconjuntos de redes que trabajan con imágenes. Sin embargo, hoy en día es una de las redes más sencillas.

ception de Christopher Nolan con Leonardo Di Caprio). Esta arquitectura de 22 capas con parámetros 5M se llama Inception-v1. Aquí, el enfoque Network in Network se utiliza mucho, esto se hace mediante "módulos de inicio". El diseño de la arquitectura de un módulo Inception es producto de la investigación sobre las estructuras dispersas. De acuerdo con el paper presentado por Szegedy, C. y su equipo en [36], cada módulo presenta tres ideas clave:

- Tener torres paralelas de convoluciones con diferentes filtros, seguidas de concatenación, captura diferentes características en 1×1, 3×3 y 5×5, por lo que las agrupa. La idea sugiere una construcción de capa por capa en la que se deben analizar las estadísticas de correlación de la última capa y agruparlas en grupos de unidades con alta relación.

- Las convoluciones 1×1 se usan para la reducción de dimensionalidad para eliminar los cuellos de botella computacionales.

- Debido a la función de activación en la convolución 1×1, su adición también agrega la no linealidad.

Vale la pena señalar que el punto fuerte de esta arquitectura es la mejor utilización de los recursos informáticos dentro de la red. Y como nos explican Calvo, R.A. & D' Mello, S. en [24], posteriormente aparece Inception-v3, como el sucesor de Inception-v1, ahora con parámetros de 24M (el v2 es el prototipo anterior al v3, pero no es muy usado ya que es muy similar

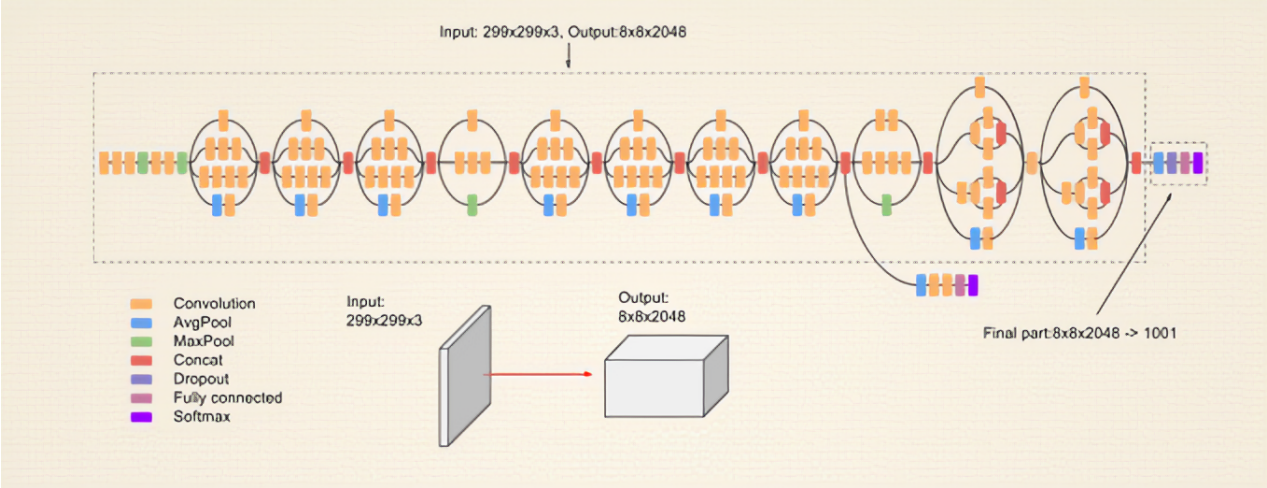


Figura 2.5: Arquitectura de la red Inception-v3. Figura obtenida de [2]

al v3). Cuando los autores lanzaron Inception-v2, realizaron muchos experimentos y registraron algunos ajustes exitosos. Inception-v3 es la red que incorpora éstos ajustes, ajustes de optimizador, función de pérdida y adición de normalización por lotes a las capas auxiliares en la red auxiliar.

La motivación para Inception v2 y v3 es evitar los cuellos de botella representacionales, lo que significa reducir drásticamente las dimensiones de entrada de la siguiente capa, y tener cálculos más eficientes mediante el uso de métodos de factorización.

2.0.8. ResNet

Las redes neuronales tipo ResNet (*residual network*) son un tipo particular de redes convolucionales. La arquitectura de las ResNet fue introducida por primera vez en 2015 por Xie, S. y colaboradores en el artículo "*Deep*

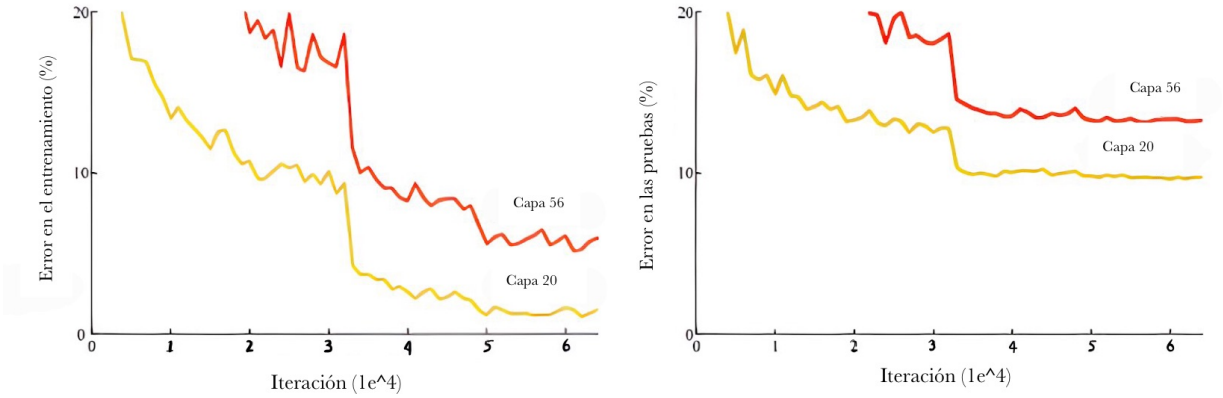


Figura 2.6: Se muestra el error de entrenamiento (izquierda) y el error de prueba (derecha) en una red de 20 capas y 56 capas. La red más profunda tiene un mayor nivel de error tanto en el entrenamiento como en la prueba. Figura traducida obtenida de [5]

Residual Learning for image recognition", [5] y es una de las arquitecturas más usadas en nuestros días. Los desarrollos más recientes en modelos de imágenes casi siempre usan el mismo truco de conexiones residuales, y la mayoría de las veces, son solo una modificación del ResNet original.

Calvo, R.A. & D'Mello, S. explican en [24] que una red tipo ResNet hace referencia a una red VGG19, se modifica en base a ella y la unidad residual se modifica basándose un mecanismo de saltos. ResNet agrega pues, un mecanismo de "salto" cada dos o tres capas de la red ordinaria, lo que conforma el aprendizaje residual.

Una razón de peso para saltar capas es evitar los gradientes de retropropagación o desvanecimiento, como el gradiente se retropropaga a las capas anteriores este proceso repetido puede hacer que el gradiente sea extremadamente pequeño [25, 30]. Mientras entrena, estos peso se ajustan a las capas anteriores y amplían la capa saltada anteriormente, a veces, los

pisos utilizados para conectar las capas adyacentes entran en juego pero esto sólo pasa si hay linealidad entre las capas.

Como se explica en el artículo de Xie, S. y colaboradores, [5], el salto elimina las complicaciones de la Red. Y usando menos capas en el entrenamiento inicial, se acelera el aprendizaje hasta 10 veces, minimizando la desaparición de los gradientes. Después de esto la Red neuronal eventualmente vuelve a poner las capas hábiles mientras aprende el espacio de características. A medida que el entrenamiento se acerca a su fin, se acelera el aprendizaje. Una Red neuronal que no tiene partes residuales, tiene más libertad para explorar el espacio de características por lo que se vuelve altamente peligroso cuando se trata de perturbaciones pues se sale del colector de características o Max-Pool. A medida que avanza el entrenamiento, el modelo capta el concepto de retener las capas útiles y no usar las capas que no ayuden. El modelo convierte las últimas capas en mapas de identidad. Es un factor importante para el éxito de la Red neuronal residual pues resulta muy sencillo crear mapas de capas para la función identidad.

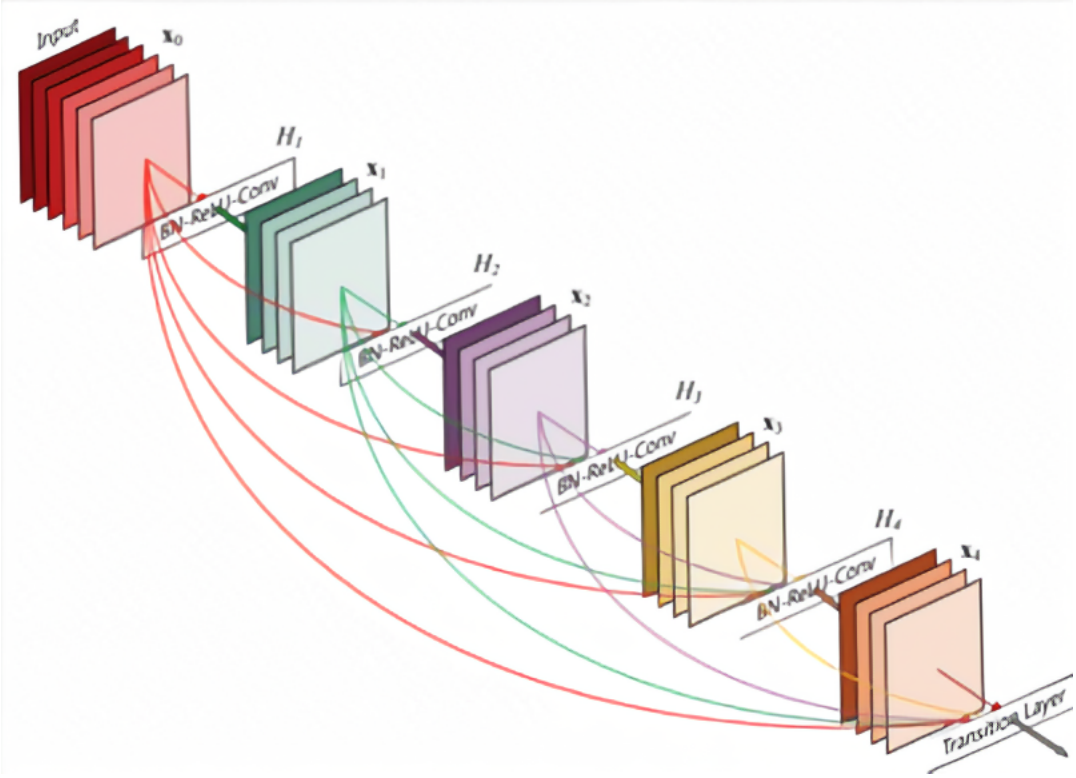


Figura 2.7: Las redes neuronales pueden ensamblar construcciones de aprendizaje usando atajos o líneas de salto sobre las capas. Figura obtenida de [2]

2.0.9. El problema de la profundidad y la retropropagación

Según Petersen, E. y colaboradores en [30], redes anteriores a las ResNet, poseen una profundidad promedio de entre 16 y 30 capas, ya con éste nivel de profundidad es posible analizar archivos de imagen hasta el high deep, que analiza objetos. Lógicamente, podríamos pensar que a mayor número de capas, mayor es el nivel de profundidad alcanzado por el modelo de red neuronal. Sin embargo, una de las principales aportaciones del primer artículo de ResNet nos dice que eventualmente, conforme crece el número

de capas de un modelo, el error presente tanto en el entrenamiento como en la prueba se ve incrementado.

De este modo, lo que se propone en el artículo publicado por Xie, S. y colaboradores del 2015, es comenzar con una red neuronal de 20 capas bien entrenada, luego agregar 36 capas que no hacen nada realmente (podrían estar alienadas con un solo peso igual a 1 y bias igual a 0). El resultado es una red neuronal de 56 capas que hace lo mismo que una red de 20, demostrando que siempre hay redes profundas que deberían ser tan buenas como cualquier red superficial.

En realidad, hay otra forma de crear esas 36 capas adicionales, que es mucho más interesante. ¿Qué pasaría si reemplazáramos cada ocurrencia de $conv(x)$ con $x+conv(x)$, donde conv es la función de la sección anterior que añade una segunda convolución, luego una ReLU, luego una capa batch-norm? De acuerdo con Talo, M. et. al. [32], nuestra $conv(x)$ para esas 36 capas adicionales siempre será igual a cero, lo que significa que $x+conv(x)$ siempre será igual a x. De este modo, esas 36 capas adicionales, tal como están, son un mapeo de identidad, pero tienen parámetros, lo que significa que son entrenables. Así que podemos comenzar con nuestro mejor modelo de 20 capas, agregar estas 36 capas adicionales que inicialmente no hacen nada y luego ajustar todo el modelo de 56 capas. Esas 36 capas adicionales pueden luego aprender parámetros, lo que las hará mucho más útiles.

El artículo de ResNet propuso una variante de esto, que consiste en ha-

cer pequeños saltos cada segunda convolución, por lo que efectivamente obtenemos $x + conv2(conv1(x))$. Los bloques residuales son el componente básico de una ResNet, en el bloque residual, la entrada X se agrega directamente a la salida de la red. Es decir, si la salida de una red lineal como la VGG-16 está representada por la función $f(x)$, entonces la salida de la ResNet estará representada por la función de salto

$$H(x) = f(x) + x \tag{2.16}$$

De modo que cuando tenemos $f(x) = 0$ en consecuencia, $H(x) = x$. A lo anterior se le conoce como mapeo identidad y en otras palabras, indica que la entrada de la red equivale a la salida de la misma. El mapeo de identidad creado por los bloques residuales es la razón por la cual la adición de capas no afecta al rendimiento de una ResNet. La mejora del rendimiento se logra siempre que las capas adicionales obtienen información significativa de los datos. Así mismo, la presencia de los bloques residuales evita la pérdida de rendimiento siempre que las activaciones aparezcan o expiren.[5]

CAPÍTULO 3

METODOLOGÍA

El primer paso para todo trabajo en machine learning es la construcción de la base de datos y su exploración. En nuestro caso, la base de datos se obtuvo mediante la plataforma Kaggle y se puede consultar en [8]. La base de datos obtenida de la plataforma consta de $19,845$ imágenes en total. Cada imagen viene etiquetada con una de cuatro catogorías o clases (COVID, Normal, Opacidad Pulmonar, Neumonía Viral). Del total de imágenes; 9,862 corresponden pacientes sanos, 1,015 denotan pulmonía viral, 4,252 pertenecen a pacientes COVID y 5,682 son de personas que por diversos motivos desarrollaron opacidad pulmonar.

Adicionalmente, se logró contactar con el Dr. Malco, médico radiólogo

que labora en el hospital MAC en la ciudad de Aguascalientes. El Dr. Malco coolaboró conmigo para proporcionarme las radiografías de 64 pacientes que fueron diagnosticados con COVID-19 mediante pruebas de PCR y que fueron internados para recibir tratamiento en el mencionado hospital. Los estudios radiológicos de los 64 pacientes fueron añadidas a la base de datos en el grupo de prueba como parte del proceso de validación cruzada. Es de destacar que la colaboración con el Dr. fue de suma importancia para el trabajo, dado que el hecho de que todas las radiografías fueron realizadas después de haberse confirmado la presencia del virus mediante la prueba PCR, nos ayuda a evaluar el verdadero desempeño del modelo.

El modelo propuesto usa como base la librería **PyThorch** para **Python**, pues sabemos que con funciones ya predeterminadas en la librería, es posible ocupar de manera directa algunas redes neuronales preentrenadas de tipo CNN. De este modo, todas las funciones en el modelo están descritas por la librería Pythorch. El modelo se corrió de manera remota en Google Colaboratory con computadoras que poseen tarjeta gráfica (GPU). Las especificaciones de las GPU usadas se muestran en la figura 3.1

```
Thu May 19 20:28:26 2022
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 460.32.03    Driver Version: 460.32.03    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla P100-PCIE...  Off  | 00000000:00:04.0 Off |                    0 |
| N/A   35C    P0    26W / 250W |      0MiB / 16280MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Figura 3.1: Especificaciones técnicas sobre el GPU y la versión de Cuda utilizada.

3.1. Arquitectura del modelo propuesto

En términos generales, la arquitectura del modelo tiene la estructura mostrada en el diagrama de flujo de la figura 3.2 y que se especifica a continuación:

1. **Extracción de las imágenes.**

   Primero se hacen pruebas para extraer los componentes de la base de datos, todos los elementos a utilizar cuentan con una etiqueta que indica la categorización individual, es decir, si es una radiografía perteneciente a un paciente sano, COVID, con neumonía viral o con opacidad pulmonar. Todos los elementos de la base de datos se ordenaron de acuerdo a la etiqueta con el diagnóstico verdadero en carpetas

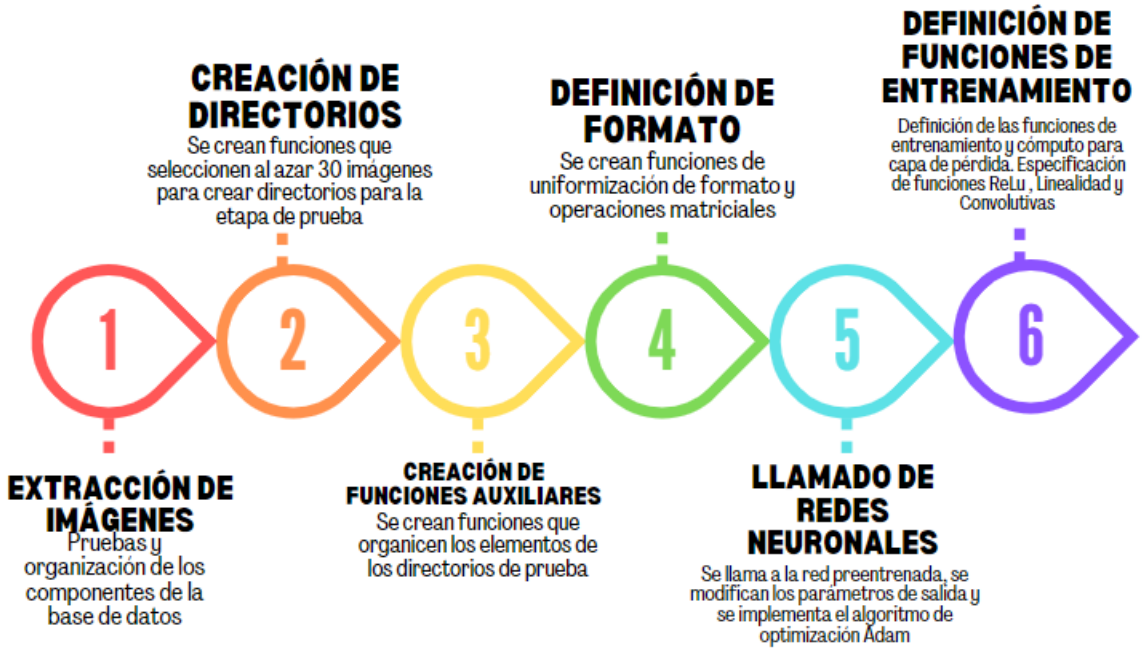


Figura 3.2: Diagrama de flujo del modelo propuesto

ubicadas en Google Drive. En la figura 3.3 se muestra una captura de pantalla ilustrando este paso.

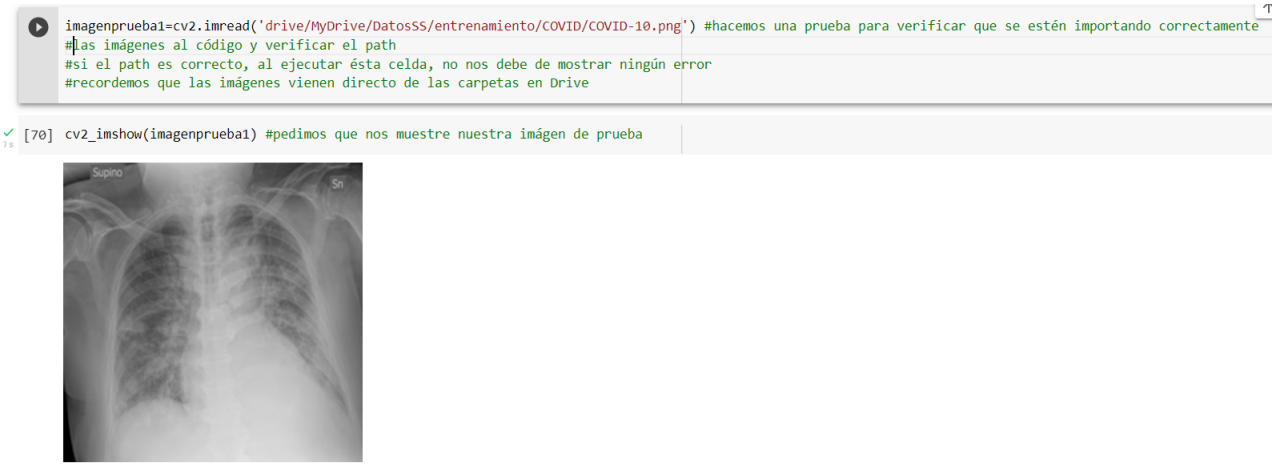


Figura 3.3: Captura de pantalla con una imagen de la base de datos importada al notebook con el modelo

2. **Creación de directorios**

   Posteriormente se procede a seleccionar al azar 30 imágenes de cada una de las categorías correspondientes y con ellas se crean directorios de prueba y entrenamiento para cada una de las categorías. Este procedimiento se realiza dos veces, uno para el conjunto de prueba y la segunda para el conjunto de entrenamiento.

3. **Creación de las funciones auxiliares**

   A continuación, se crean las funciones auxiliares que nos serán de utilidad para crear los directorios y clases que se usarán en la etapa de entrenamiento. Dentro de una nueva clase se definen 3 funciones. La función `len` sirve como punto de referencia para conocer el número de elementos que se están introduciendo en la clase. Por su parte, la función `get item` nos ayuda para obtener al azar elementos de los directorios que hemos creado en etapas anteriores. Finalmente, la función `init` es la función que inicializa el proceso de entrenamiento llamando a las otras dos funciones.

4. **Definición de funciones para uniformizar el formato de imagen**

   Por el momento, el modelo está trabajando con un banco de imágenes en formato pdf, todas las imágenes con dimensiones relativamente similares. Sin embargo, se espera que el usuario introduzca sus propias imágenes. Por lo que resulta necesario uniformizar el formato, la orientación y las dimensiones de las imágenes.

Ya que se han preparado las imágenes, las pasamos a vector mediante la función `ToTensor` de la librería **TorchVision** además, normalizamos el vector con la función `Normalize` también de la librería **TorchVision**.

5. **Llamado y preparación de la Red Neuronal**

   Como ya he mencionado, mediante la librería **PyThoch**, es posible llamar redes neuronales preestablecidas. Así, con el objetivo de realizar experimentos de funcionabilidad y optimización se utilizan 5 redes. Las redes utilizadas son; $ResNetxt101$, $ResNet50$, $ResNet18$, $InceptionV3$ y $VGG16$. Cabe mencionar que, todas las redes fueron modificadas para su uso en el modelo y además, para las redes de la familia $ResNet$ se implementa la optimización $AdamNet$.

6. **Definición de las funciones de entrenamiento**

   Para la etapa del entrenamiento de la red, comenzamos por definir el tamaño de los lotes de imágenes y definimos la función de entrenamiento que introduce el lote en la red y analiza el desempeño de la misma, se usaron los recursos expuestos en la sección introducción, incluidas funciones en la capa de pérdida y especificaciones en las funciones de linealidad. El output de la función es la predicción correspondiente como se muestra en la figura 3.4. Recordemos que todas las redes utilizadas pertenecen a la categoría de modificación matricial y, como se menciona en el punto 4, las imágenes ya están transformadas en vectores normalizados antes de entrar a la red.

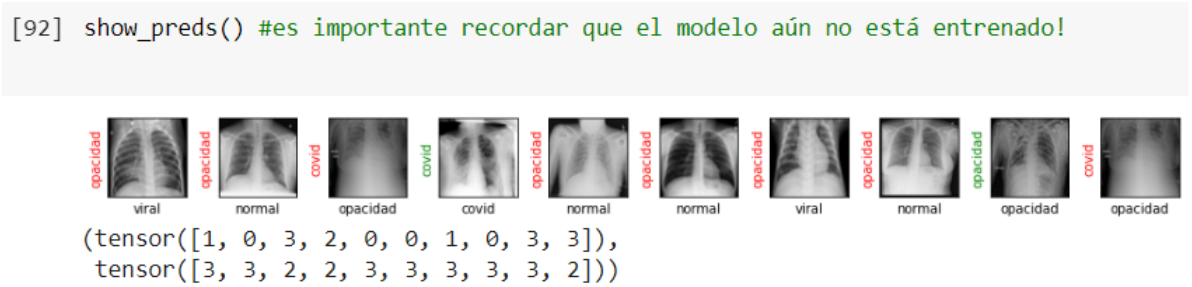


Figura 3.4: Tras definidas las funciones auxiliares, se define el tamaño de los lotes y se define una nueva función que muestre las predicciones del modelo. Los tensores que se muestran en la parte inferior de la imagen corresponden con los tensores que contienen las etiquetas asignadas por el modelo para el directorio de cada clase.

En este punto también se definen funciones que nos ayuden a evaluar el desempeño del modelo con las distintas redes neuronales, una de las funciones más importantes es la función `loss` o función pérdida, la cuál se computa a partir de la entropía de Shannon. El método utilizado se especifica con mayor detalle en la sección 3.2.

Es importante destacar que, como se menciona anteriormente, la función que hemos definido también evalúa el desempeño del entrenamiento de la red pues conforme se dan las predicciones, la función de entrenamiento compara el resultado con el etiquetado inicial de las imágenes. Mediante ésta función podremos evaluar el desempeño del modelo con las diferentes redes neuronales de acuerdo a las métricas de evaluación que se describen en la siguiente sección.

3.1.1. Métricas de evaluación para evaluaciones de clasificación de datos.

En lo que refiere a modelos para la clasificación de datos podemos encontrar diferentes enfoques como son la clasificación binaria, multiclase y multietiquetada. El modelo desarrollado para la realización de este trabajo realiza una clasificación multiclase, en años recientes se ha popularizado este tipo de clasificación, por lo que las métricas comúnmente utilizadas para evaluar modelos de clasificación binaria se han generalizado para casos del tipo multiclase. De este modo, partiremos de las métricas para la evaluación de modelos de clasificación binaria.

Las métricas de evaluación para el aprendizaje automático nos permite alcanzar transparencia en la lectura de los resultados del método desarrollado aún para aquellos que no están familiarizados con los temas de aprendizaje profundo. Las métricas de evaluación pueden categorizarse en 3 tipos; límite, métrica de probabilidad y clasificación.

Las afirmaciones sobre la eficacia de los algoritmos de aprendizaje automático a menudo detallan la calidad del algoritmo utilizando un conjunto básico de medidas de rendimiento. Generalmente, los modelos que clasifican la ausencia o presencia de alguna afección médica, como es nuestro caso, suelen ser evaluados con la curva ROC o AUROC (área bajo la curva) y la llamada matriz de confusión [37] [29].

Matriz de confusión y métricas de evaluación tipo límite

Para problemas de clasificación binaria, la evaluación por discriminación suele ser la opción más óptima en lo que refiere a la etapa de entrenamiento. Durante la clasificación de prueba, se puede definir una matriz de confusión como la que se muestra en la figura 3.5.

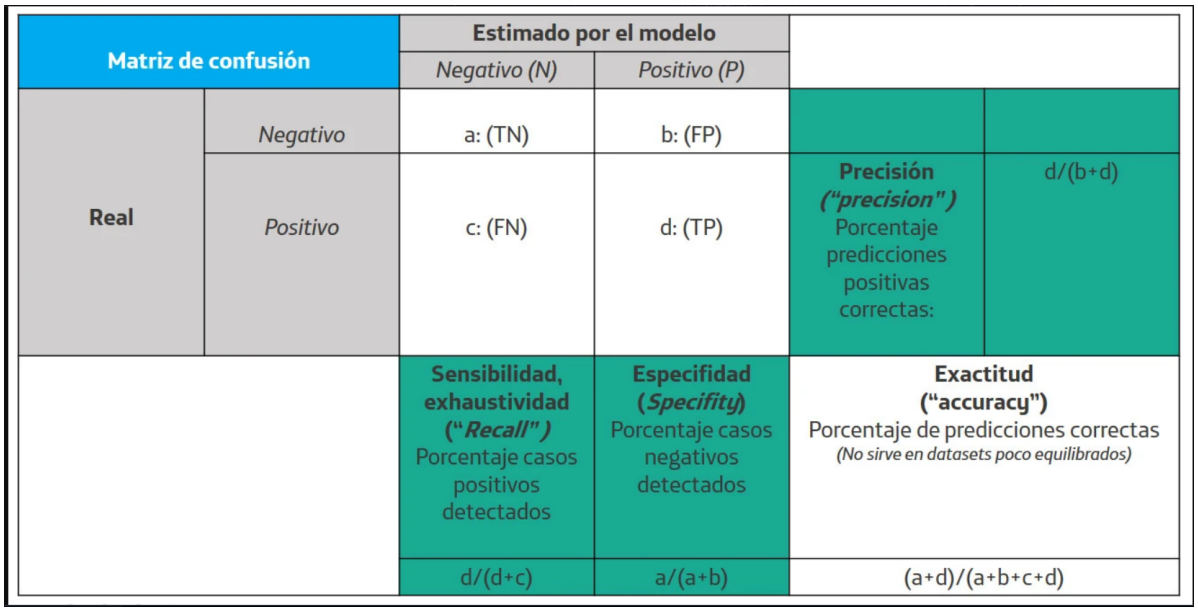


Figura 3.5: Matriz de confusión para un clasificador binario.

La exactitud es la métrica de evaluación más usada tanto en clasificaciones binarias como de tipo multiclase. A través de la precisión podemos entender la calidad de la solución producida basada en el porcentaje de predicciones correctas sobre las totales. Por su parte, tenemos que la métrica complementaria de exactitud es la tasa de error o pérdida que evalúa la solución producida por su porcentaje de predicciones incorrectas. Ya hemos definido la función pérdida mediante la ecuación (2.15). En la tabla 3.1 se enlistan algunas de las métricas de evaluación más utilizadas en los

Tabla 3.1: Métricas de evaluación de clasificaciones tipo límite.

Métricas	**Fórmula**	**Enfoque de evaluación**
Exactitud (acc)	$\frac{TP+TN}{TP+FP+TN+FN}$	Mide la razón de predicciones correctas entre el total de elementos evaluados.
Rango de error (err)	$\frac{FP+FN}{TP+FP+TN+FN}$	Mide la razón de predicciones incorrectas entre el total de elementos evaluados
Sensitividad (sn)	$\frac{TP}{TP+FN}$	Mide la razón entre patrones positivos y los correctamente clasificados
Especificidad (sp)	$\frac{TN}{TN+FN}$	Mide la razón entre los patrones negativos y los correctamente clasificados
Precisión (p)	$\frac{TP}{TP+FP}$	La precisión se utiliza para medir los patrones positivos que se pronostican correctamente a partir del total de patrones pronosticados en una clase positiva.

problemas de clasificación binaria y de tipo multiclase. Mediante la comparación de verdaderos positivos y verdaderos negativos en una muestra de prueba con los datos positivos previstos y negativos previstos, se puede calcular una variedad de estadísticas de resumen. En la tabla 3.1 y la figura 3.5 podemos encontrar la descripción de algunas de las estadísticas más socorridas para la evaluación de modelos. La tasa de verdaderos positivos (TP por sus siglas en inglés) también conocida como recuento o sensibilidad, hace referencia a la probabilidad de que el modelo detecte un caso positivo con veracidad. Por su parte, la tasa de falsos negativos (FN por sus siglas en inglés) es la proporción de casos positivos perdidos. Ahora

bien, la tasa de falsos positivos (FP), es la probabilidad de identificar incorrectamente un caso como positivo cuando la condición no está realmente presente. Y finalmente, la tasa de verdaderos negativos (FN), también conocida como especificidad, es la probabilidad de acertar al descartar la condición. Y tal como se aprecia en la tabla 3.1, los valores anteriormente descritos nos servirán para encontrar las métricas de evaluación tipo límite de nuestro modelo y pueden ser calculadas a partir de la matriz de confusión. Notemos que para los modelos de clasificación multi clase, la matriz de confusión varía con respecto a la matriz de un caso binario. Para ejemplificar la generalización de la matriz de confusión para clasificadores multiclase, supongamos que tenemos un modelo con 4 clases (A, B, C, D). De este modo, la matriz de confusión para nuestro modelo hipotético se muestra en la tabla 3.2. Notemos que la diagonal de la matriz corresponde con los valores positivos verdaderor (TP) para cada clase y el resto de las entradas corresponden con valores de error. Por ejemplo, la entrada E_{BA} corresponde con una entrada clasificada como A siendo que su clase real era B.

Además, también de la tabla 3.2, podemos notar que el número de falsos negativos (FN) por clase será la suma de todas las entradas en la fila correspondiente a la clase exceptuando el valor positivo verdadero (TP). Análogamente, el falso positivo (FP) para una determinada clase será la suma de las entradas de su fila exceptuando el valor positivo verdadero. Finalmente, el negativo verdadero (TN) de una clase equivale a la suma

de todas las entradas de la matriz excepto por las que se encuentran en la fila y columna correspondientes a la clase en cuestión, es decir, el negativo verdadero correspondiente para la clase B es

$$TN_B = TP_A + E_{AC} + E_{AD} + E_{CA} + TP_C + E_{CD} + E_{DA} + E_{DC} + TP_D \quad (3.1)$$

Por lo tanto, en su forma básica, la matriz de confusión es una representación de términos como la sensibilidad y la especificidad, permitiendo a los lectores interesados en una métrica específica para un algoritmo en particular, consultar rápidamente y comparar con otros algoritmos fácilmente.

Tabla 3.2: Matriz de confusión generalizada para clasificadores no binarios.

		Valores Predichos por el modelo			
	Clases	A	B	C	D
Valores reales	A	TP_A	E_{AB}	E_{AC}	E_{AD}
	B	E_{BA}	TP_B	E_{BC}	E_{BD}
	C	E_{CA}	E_{CB}	TP_C	E_{CD}
	D	E_{DA}	E_{DB}	E_{DC}	TP_D

De las métricas descritas en la tabla 3.1, podemos destacar un valor que suele citarse en diversos estudios publicados. La precisión es el número de predicciones correctas realizadas como proporción de todas las predicciones realizadas, una alta precisión nos habla de un buen rendimiento. Sin embargo, debemos notar que un modelo para detectar la presencia o ausencia de una embolia pulmonar significativa en angiografía pulmonar con

precisión del 90 %, es erróneo en el 10 % de los casos. Destaquemos que identificar incorrectamente una embolia pulmonar como ausente es probablemente peor que decir que está presente. Por lo tanto, dependiendo de la pregunta clínica, puede ser más importante considerar las otras métricas de la matriz de confusión, como el valor de probabilidad negativa, que informa al médico más directamente que la precisión sobre qué tan confiable es un resultado de prueba negativo o una predicción.[38]

Validación cruzada

En general y de acuerdo con Youden, W.J. en [39], si un investigador o desarrollador desea comprobar si su algoritmo es funcional, es necesario probarlo en una población concreta. Es decir, si tenemos una población o base de datos con 1,000 sujetos de estudio; podemos tomar 800 sujetos de la población inicial como grupo de entrenamiento para el modelo predictivo y usar a los 200 restantes como sujetos de prueba para el modelo. Cada iteración no solo mejorará el rendimiento del modelo, ya que el programa puede comparar los resultados de cada conjunto de entrenamiento para ver qué funciona mejor y puede alterar su capacidad predictiva general, sino que también mejorará la generalización de los resultados. A medida que un algoritmo trata con muchas combinaciones de pacientes, la posibilidad de sobreajustar el algoritmo predictivo disminuye. Podremos ver que al separar los datos en categorías, predicciones correctas e incorrectas, es posible sobre ajustar el modelo a los datos de muestra, lo que hace que

la generalización del modelo a la población no sea realista. Análogamente, al hacer que la generalización del modelo sea muy generalizable, es posible ajustar el modelo y que las predicciones sean débiles. Este proceso de validación a través de un conjunto de pruebas teóricas permite la representación numérica del rendimiento de un algoritmo porque puede ver cuántas predicciones fueron correctas e incorrectas.

3.2. Entropía Cruzada

Como físicos y científicos, entendemos el importante papel de la entropía, en los últimos años, el uso de la entropía no solo ha llevado a un aumento significativo en la comprensión de sistemas físicos, si no en el estudio de la complejidad contribuyendo incluso al surgimiento de nuevas disciplinas científicas.

Sabemos que el concepto de —entropía física surge de la revolución industrial, donde la observación de que en las máquinas de vapor se perdía gran parte de la energía debido a la fricción y la disipación y, por lo tanto no podía convertirse en trabajo útil. R. Clausius presentó la primera expresión matemática para describir la entropía en 1854 como:

$$S = \frac{q}{T} \tag{3.2}$$

o bien

$$\Delta S = \left(\frac{1}{T_2} - \frac{1}{T_1}\right) * q \tag{3.3}$$

Dónde ΔS representa cambios en la entropía, mientras que q es la cantidad de calor que pasa del cuerpo con la temperatura T_1 a otro cuerpo con temperatura T_2.

Tiempo después, Boltzmann formuló una definición ligeramente diferente de entropía, siendo una medida del desorden molecular del sistema. Tiene la siguiente forma:

$$S = k_B \ln(W) \tag{3.4}$$

Donde k_b representa la constante de Boltzmann y W es el número total de estados microscópicos del sistema. La entropía de Boltzmann. La entropía de Boltzmann proporciona la base para la mecánica estadística, pero el concepto de probabilidad, que es crucial en la teoría estadística, no resulta directamente de ella. Sabemos que la probabilidad termodinámica de la ecuación (3.4) no es una probabilidad ordinaria, si no un número entero. Sin embargo, si un sistema aislado que consta de N moléculas perteneciente a n estados energéticos es examinado, suponiendo un número fijo de moléculas y valores totales de energía, el estado total de estados microscópicos totales está dado por:

$$W = \frac{N!}{\prod_{i=1}^{n} N_i!} \tag{3.5}$$

De este modo, sustituyendo (3.5) en (3.4), obtenemos una nueva expresión para la ecuación de Boltzmann, que es:

$$S = k_B \ln \frac{N!}{\prod_{i=1}^{n} N_i!} \approx -k_B N \sum_{i=1}^{n} p_i \ \ln \ p_i \tag{3.6}$$

dónde $p_i = N_i/N$ para un N grande. Lo cual implica la probabilidad de que la molécula se encuentre en el i-ésimo estado de energía.

Luego, a mediados del siglo XX, el concepto de entropía encontró su aplicación en áreas informáticas. Veremos que la entropía informática resulta muy similar a la entropía termodinámica. Para dos variables discretas aleatorias discretas, p y q, la entropía cruzada mide la media de bits necesarios para identificar un evento de un conjunto de posibilidades [40]. En 1948, Claude E. Shannon publicó su trabajo, *A Mathematical Theory of Communication* donde se definió la entropía de Shannon o energía cruzada de la siguiente manera

$$H = -K \sum_{i=1}^{n} p_i \ \log \ p_i \tag{3.7}$$

tal que K representa una constante positiva. Podemos observar facilmente que las (3.6) y (3.7) se parecen mucho. H representa la entropía conocida en mecánica estadística, donde p_i denota la probabilidad de que un sistema se encuentre en la celda i dentro de su espacio de fase. La entropía de Boltzmann es propocional a la entropía de Shannon. Y al mismo tiempo, resulta importante enfatizar como las ecuaciones en esta sección son todas homologías lógicas ya que presentan la misma estructura formal matemática en varios niveles de realidad. Particularmente, la entropía de Shannon puede ser tratada como un concepto más general de la entropía termodinámica estadística. De esta forma, la predicción de las propiedades termodinámicas de equilibrio puede proporcionar una forma de inferencia estadística basada en la entropía de Shannon como medida de información, mientras

que las probabilidades se interpretan de manera subjetiva.

De este modo, entendemos que la entropía de Shannon o entropía cruzada es una métrica que puede utilizarse para reflejar la precisión de los pronósticos probabilísticos y está estrechamente vinculada con la estimación por máxima verosimilitud. Es por esto que la entropía cruzada es de gran importancia en los sistemas pronósticos modernos. Particularmente, en modelos clasificadores como el propuesto en esta tesis.

Cuando tenemos un modelo clasificador con C clases, se espera que la entrada de una función que compute la pérdida del sistema a partir de la entropía de Shannon, contenga puntajes sin procesar y no normalizados para cada clase. La entrada se supone como un tensor de tamaño C y el objetivo es que se obtenga el resultado para la expresión de pérdida dada por

$$l(x, y) = L = l_1, ..., {l_N}^T \tag{3.8}$$

que, de acuerdo con la ecuación (3.7), es

$$l_n = -\sum_{c=1}^{C} w_c \log \frac{exp(x_n, c)}{\sum_{i=1}^{C} exp(x_{n,i} y_{n,c})} \tag{3.9}$$

dónde N es la dimensión de los lotes, x es la entrada, y es el target, w es el peso, C es el número de clases.

CAPÍTULO 4

RESULTADOS

Tal como se mencionó en secciones anteriores, se realizaron pruebas del modelo desarrollado con varias redes neuronales. En las imágenes de la figura 4.1 podemos encontrar capturas de pantalla del modelo en funcionamiento durante la etapa de entrenamiento. Se escogieron lotes de 10 imágenes para facilitar el cálculo. En las imágenes de la figura 4.1 podemos apreciar como el modelo no detiene el entrenamiento hasta alcanzada la condición de satisfacción.

En la tabla 4.1 se encuentran las exactitudes alcanzadas durante la etapa de prueba del modelo propuesto entrenado con las diferentes redes neuronales. Recordemos que la exactitud es la métrica que expresa la cercanía del valor de una o más mediciones independientes a un valor verdadero.

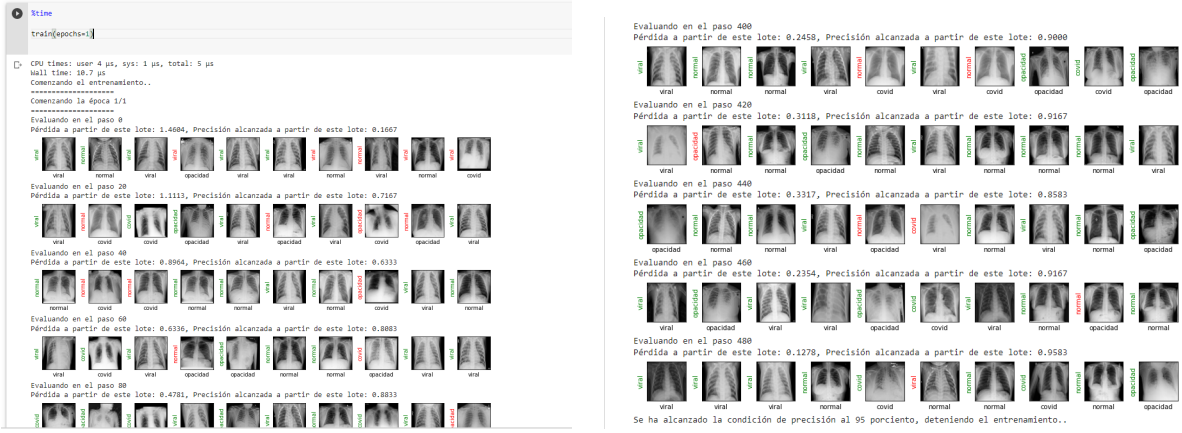


Figura 4.1: Capturas de pantalla del modelo al iniciar (derecha) y terminar (izquierda) la etapa de entrenamiento mediante la red neuronal Inception V3.

Entre más cerca se encuentra una medición del valor real, menor es el error y más exacta es. La expresión que define la exactitud se encuentra en la tabla 3.1 en la sección de metodología.

En la segunda columna de la tabla 4.1 se encuentran las exactitudes globales alcanzadas por el modelo con las redes neuronales de la primer columna. Así mismo, en la última columna se encuentran las exactitudes alcanzadas en cada categoría. Todas las exactitudes se calcularon usando la expresión para exactitud contenida en la tabla 3.1, que se encuentra en la metodología, sólo que en las globales se usó el total de las imágenes evaluadas y en el caso contrario se calcularon usando los totales por clase.

En general, encontramos que las exactitudes globales resultan superiores al 94 %, mientras que la exactitud por clase más baja por categoría resulta

Tabla 4.1: Exactitudes obtenidas durante la etapa de prueba del modelo entrenado con 5 redes neuronales diferentes.

Red Neuronal usada	Exactitud global	Clase	Exactitud alcanzada por clase
VGG16	0.95	COVID	0.94
		Normal	0.89
		Neumonía Pulmonar	0.99
		Opacidad Pulmonar	0.93
ResNet18	0.97	COVID	0.97
		Normal	0.94
		Neumonía Pulmonar	1
		Opacidad Pulmonar	0.97
ResNet50	0.95	COVID	0.98
		Normal	0.89
		Neumonía Pulmonar	0.98
		Opacidad Pulmonar	0.93
ResNetXt101	0.94	COVID	0.97
		Normal	0.87
		Neumonía Pulmonar	0.97
		Opacidad Pulmonar	0.93
Inception V3	0.98	COVID	0.96
		Normal	0.99
		Neumonía Pulmonar	1
		Opacidad Pulmonar	0.97

ser del 87 %. La red con mayor exactitud global fue la red Inception V3 con un 98 %. Hablando de las categorías individualmente, la red con mayor exactitud alcanzada para la clase COVID fue ResNet 50, tal que se alcanzó

una exactitud del 98 %. En el caso de la clasificación Normal y Neumonía Pulmonar, los mejores resultados se alcanzaron utilizando la red Inception V3 con exactitudes del 99 % y 100 % respectivamente. Finalmente, en el caso de la categoría Opacidad Pulmonar ocurrió un empate entre las redes ResNet18 e Inceptión V3 con una exactitud del 97 %.

4.1. Red VGG16

Respecto a modelos que precedieron a la primer red VGG, una característica revolucionaria para su tiempo fue la primer capa convoluciónal, que proponía el uso de un campo receptivo muy pequeño de 3×3 en toda la red con paso de un píxel. Lo que implica un filtro más pequeño en comparación con el tamaño del campo receptivo de redes previas, por ejemplo, el campo receptivo de la AlexNet era de 11×11 con paso de 2 píxeles. De este modo, aunque la arquitectura de la red VGG puede parecer simple, la combinación de múltiples filtros tipo 3×3 puede reemplazar un área receptiva de mayor tamaño, como podría ser una sola capa de 7×7. Así mismo, basándonos en una arquitectura tradicional, un campo receptivo de tamaño 3×3 se destacará sobre uno de 7×7, ya que, además de las tres capas de convolución, también habrá tres capas de activación no lineal. Por lo tanto, de acuerdo con lo visto en la sección 1, más capas convolutivas y de activación no lineal implican funciones de decisión con mayor capacidad de discriminación, por lo que en principio, la red deberá converger más rápido. Ahora, se propusieron varias configuraciones para redes tipo VGG, sin embargo, dentro de sus grupos de prueba, la VGG16 demostró ser la red con un mejor desempeño al someterse a la base de datos ImageNet [33], [35].

Como ya se ha mencionado en repetidas ocasiones, el modelo fue diseñado para continuar con la etapa de entrenamiento hasta alcanzar una

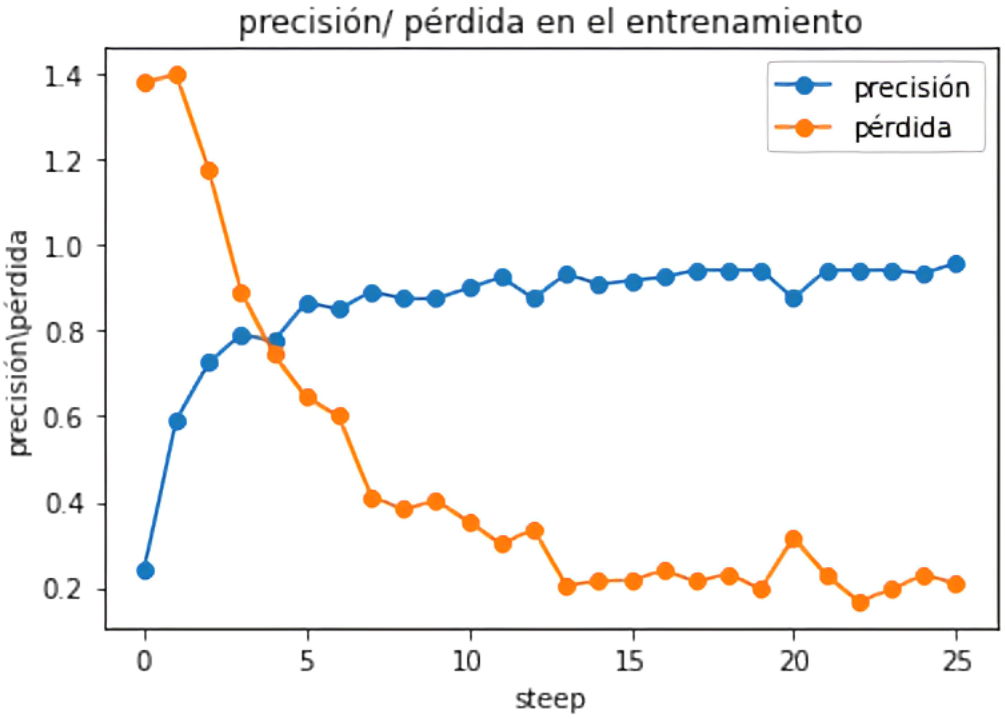


Figura 4.2: Gráfica de pérdida y precisión para durante el entrenamiento mediante la red VGG16

condición de satisfacción fijada por el usuario. En nuestro caso, se preestableció una condición del 95 % de precisión para detener el entrenamiento, lo que se traduce en una mayor o menor cantidad de pasos necesarios para alcanzar dicha condición. En la gráfica de la figura 4.2 podemos apreciar el proceso evolutivo de nuestro modelo a lo largo del entrenamiento hasta lograr la condición de satisfacción. En el eje de las abscisas podemos ver los "steeps" o pasos necesarios para llegar a la condición de satisfacción y en el eje de las ordenadas se encuentran la pérdida y precisión escaladas del 0 al 1. De este modo, cada punto de la gráfica corresponde con la precisión o pérdida alcanzados por un lote de imágenes determinado en una etapa específica del entrenamiento.

En la figura 4.2 se puede apreciar que, efectivamente, se sigue una tendencia positiva para la curva de precisión conforme avanza el entrenamiento, mostrando que a mayor número de pasos, mayor precisión alcanzamos. El comportamiento anterior resulta consistente al compararse con la tendencia de la curva de pérdidas, la cuál es propensa a disminuir con la evolución de los pasos. Es decir que ambas curvas se complementan y muestran un progreso adecuado. Sin embargo, es importante destacar que aunque la tendencia en ambas curvas es alentadora, podría ser más suave. En algunos pasos podemos observar crestas y valles, indicando retrocesos en la etapa de entrenamiento, lo que implica un mayor tiempo de entrenamiento y más exigencia para la memoria RAM del equipo utilizado. Algunos equipos de cómputo podrían no soportar entrenamientos tan largos como el que se aprecia en la gráfica de la figura 4.2, que es un punto negativo a considerar.

En la gráfica de la figura 4.3 podemos ver la misma curva de pérdidas comparada con la curva de pérdidas en la validación cruzada. Recordemos que la validación cruzada es un proceso que nos ayuda a confirmar si el comportamiento del modelo en el entrenamiento es consistente con un conjunto de prueba para cada iteración a lo largo del entrenamiento. En nuestro caso, el proceso de validación cruzada cobra especial importancia pues tenemos una base de datos desequilibrada y no muy grande en lo que respecta con los tamaños de los bancos de imágenes que se utilizan habitualmente al realizar entrenamientos de modelos de Deep Learning, y esto

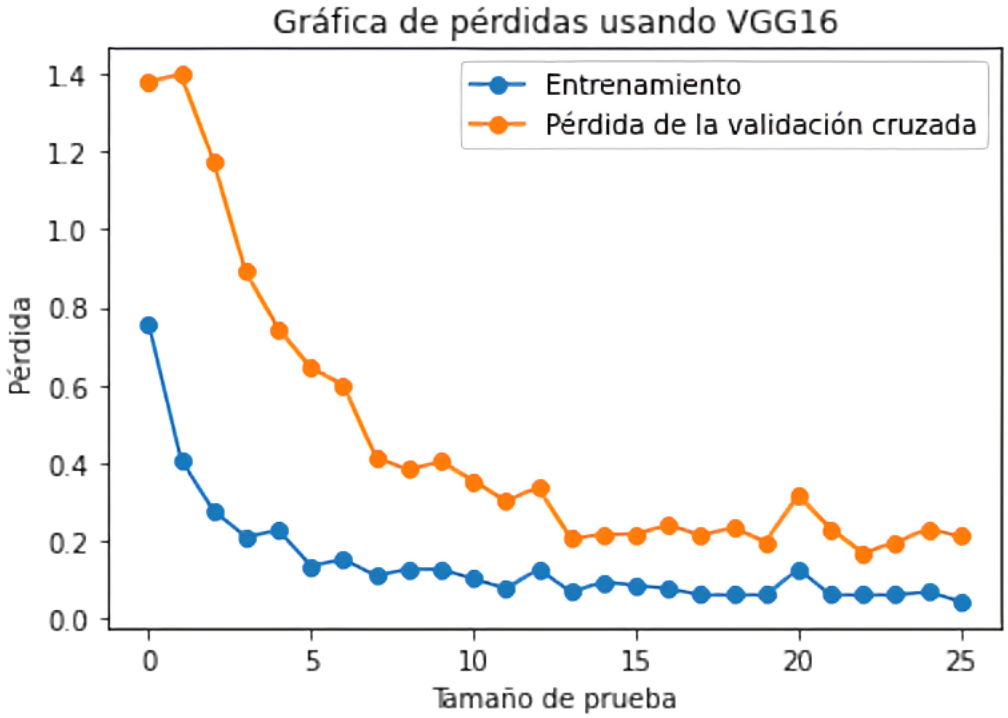


Figura 4.3: Gráfica de pérdidas con validación cruzada para VGG 16

se traduce en un alto riesgo de sobre ajustar el algoritmo predictivo. De este modo, podemos decir que el proceso de validación cruzada nos permite ver qué tan bien generalizado está el modelo.

La gráfica de la figura 4.3 nos indica una buena adaptación del modelo en general. Podemos apreciar un buen ajuste del modelo al encontrar, como ya mencionamos anteriormente, una pérdida de entrenamiento y validación que disminuye hasta un punto de estabilidad con una brecha poco significativa entre ambas curvas. Así mismo, es esperable que la pérdida del modelo en la etapa de entrenamiento se encuentre por debajo de la curva de pérdidas en la etapa de pruebas, ya que esto indica que nuestro mode-

lo tiene un aprendizaje persistente. En lo que respecta a nuestro modelo, aunque tenemos el comportamiento esperado, lo que indica un buen ajuste y generalización del mismo, no nos encontramos en el caso más óptimo. Debemos destacar que la brecha entre ambas curvas de pérdida es, aunque no alarmante, significativa, sobre todo en los primeros lotes de imágenes a evaluar, esto se debe al desequilibrio que existe en nuestra base de datos. De este modo, podemos concluir que, en general, nuestro modelo presenta un buen ajuste y generalización al entrenarse con la red VGG16, tal que tiene curvas evolutivas con tendencias positivas y complementarias entre sí. Sin embargo, las pruebas de validación cruzada, indican que el modelo se encuentra al borde de un sobreajuste, debido al desequilibrio de la base de datos, al menos en su conjunto con esta red neuronal.

Ahora bien, ya en la etapa de prueba, lo resultados obtenidos tras el entrenamiento del modelo con la red VGG16 se encuentran en la tabla 4.2 a modo de matriz de confusión. El proceso para obtener las matrices de confusión fue tal cuál se manifestó en la sección de metodología (3) y se realizó para un conjunto de 100 imágenes seleccionadas al azar bajo criterios de homogeneidad en el conjunto, es decir, procurando obtener muestras homogéneas de cada clase a evaluar.

De los resultados obtenidos en la matriz de confusión contenida en la tabla 4.2 y las definiciones de la tabla 3.1, obtenemos los valores que se expresan en la tabla 4.3. En general, obtenemos grupos ordenados por clase

Tabla 4.2: Matriz de confusión para el modelo entrenado mediante la red neuronal VGG16

Matriz de confusión para VGG16					
		Valores predichos por clase			
		COVID	Normal	Neumonía Viral	Opacidad Pulmonar
Valores reales	COVID	28	3	0	1
	Normal	2	18	1	1
	Neumonía Viral	0	0	20	0
	Opacidad Pulmonar	0	2	0	24

relativamente homogéneos, lo que resulta consistente con la brecha entre las curvas de pérdida que se mencionaba párrafos atrás. Analicemos los resultados obtenidos en cada clase.

Para la categoría COVID, encontramos que de las 100 imágenes en el conjunto de prueba, 32 estaban etiquetadas como pertenecientes a clase COVID. Encontramos una clasificación alentadora en lo que corresponde a las columnas de la parte superior de la tabla 4.3, tal que las clasificaciones positivas verdaderas (TP) y negativas verdaderas (TN) fueron mayoritarias en contraste con las falsas. Lo anterior se refleja en los resultados de precisión, especificidad, sensibilidad y rango de error.

Recordemos que la precisión se refiere a la dispersión del conjunto de valores obtenidos a partir de mediciones repetidas de una magnitud. De la expresión en la tabla 3.1 para la precisión podemos decir que cuanto menor

Tabla 4.3: Tabla resumen de resultados para VGG16, de acuerdo con las definiciones contenidas en 3.1

	TP	FN	Total por clase	FP	TN
COVID	28	4	32	2	66
Normal	18	4	22	5	73
Neumonía Viral	20	0	20	1	79
Opacidad Pulmonar	24	2	26	2	72
	Precisión	Sensibilidad	Especificidad	Rango de error	
COVID	0.933333333	0.875	0.970588235	0.06	
Normal	0.782608696	0.818181818	0.935897436	0.09	
Neumonía Viral	0.952380952	1	0.9875	0.01	
Opacidad Pulmonar	0.923076923	0.923076923	0.972972973	0.04	

es la dispersión mayor la precisión. Nuestro modelo obtuvo una precisión del 93 % al entrenarse con VGG16, por lo que se puede estimar al modelo como poco disperso.

Por su parte, la sensibilidad o recall, también conocida como Tasa de Verdaderos Positivos (True Positive Rate) ó TPR. Es la proporción de casos positivos que fueron correctamente identificadas por el algoritmo. De este modo, como obtuvimos una sensibilidad del 87.5 %, podemos decir que nuestro modelo resulta confiable en lo que refiere a diagnósticos positivos de la categoría COVID tras entrenarse con la red VGG16.

Finalmente, recordemos que la especificidad (también conocida como

la Tasa de Verdaderos Negativos o "true negative rate") nos habla de los casos negativos que el algoritmo ha clasificado correctamente, y expresa cuan bien puede el modelo detectar a una clase en particular. De este modo, al obtener una especificidad del 97 %, entendemos que el modelo está realizando una correcta clasificación en lo que a los casos COVID se refiere.

Ahora bien, en lo que concierne a otras clasificaciones, resulta de interés la alta sensibilidad obtenida para las clasificaciones de Neumonía Viral y Opacidad Pulmonar ya que respectivamente obtuvimos sensibilidades del 100 % y 92.3 %, lo que resulta sumamente contrastante con los resultados obtenidos en la clasificación de Neumonía Viral. De acuerdo con lo que se observa en la tabla 4.3, en la clasificación de pacientes sanos o normales, encontramos una precisión del 78.3 %, una sensibilidad del 81.8 % y una especificidad del 93.6 %. Las cifras anteriores nos indican, de acuerdo con las definiciones que hemos manejado, que el modelo está presentando una dispersión considerable, que es concurrente con la evaluación de la sensibilidad. La sensibilidad obtenida nos habla de que no se está realizando una clasificación del todo confiable de los casos verdaderos, es decir, hay una ligera desviación cuando el modelo diagnostica a un paciente como sano. También es importante mencionar que aunque la clasificación Normal no obtuvo el mejor desempeño, nos encontramos en una situación muy favorable puesto que aunque no identifica óptimamente los casos sanos, sí tiene un buen desempeño en la sensibilidad, que recordemos nos indica

una correcta detección de pacientes no saludables. Por lo tanto, el modelo ha tenido un buen desempeño en identificar pacientes no saludables y clasificarlos correctamente en alguna de las 3 patologías incluidas. Cabe mencionar además que como herramienta para un especialista, es de mayor interés que el modelo pueda alertar correctamente sobre pacientes que no están saludables y dados los resultados de la tabla 4.3, tenemos resultados muy alentadores con rango de error menor al 1 % en todas las categorías.

4.2. ResNet 18

En secciones anteriores ya hemos hablado de la motivación para llegar a la arquitectura de las redes neuronales residuales, también conocidas como ResNets. El problema de la profundidad y la retropropagación y el incremento en el error encontrado en redes neuronales con más de 25 capas, hace de las ResNets una opción de interés al evaluar el desempeño de un mismo modelo con distintas redes neuronales. Sabemos que las redes ResNets son arquitecturas de redes neuronales comúnmente utilizadas para aplicaciones de visión artificial de aprendizaje profundo, con un buen desempeño en la detección de objetos y la segmentación de imágenes. Particularmente y como es de imaginarse, la ResNet18 es una arquitectura de red con 18 capas convolucionales, que parece no muy profunda con respecto a la red anterior que posee 13 capas convolucionales, pero como ya se ha discutido anteriormente, la particularidad de una red ResNet se encuentra en su arquitectura construida a base de bloques de aprendizaje residual.

Entre más profunda es una red neuronal, más difícil será su entrenamiento. La ResNet18 representa un marco de aprendizaje residual que facilita el entrenamiento de una red que es sustancialmente más profunda de lo que es por decir, la red VGG16. En su artículo, Xie, S. et. al [5], expresan como el reformular explícitamente las capas como funciones residuales de aprendizaje con referencia a las entradas de la capa en lugar de aprender funciones no referenciadas, brinda evidencia de que este tipo de redes re-

sultan más fáciles de optimizar y pueden ganar precisión a partir de una profundidad considerablemente mayor. Esto tras evaluarse redes residuales con una profundidad de hasta 152 capas, 8 veces más profundas que las redes VGG pero aún con menor complejidad en el conjunto de datos de ImageNet.

Así, siendo una red neuronal de especial interés para la autora de este trabajo, se realizó el entrenamiento del mismo modelo pero ahora con la red neuronal ResNet18. Tal como se hizo con la red anterior y siguiendo la metodología descrita en la sección de 3, la red se ejecutó mediante la librería `PyThorch` mediante el método de aprendizaje por transferencia y se optimizó con AdamNet. Tras realizado este procedimiento, para la etapa de entrenamiento se obtuvieron las curvas de pérdida y precisión que se encuentran en la gráfica de la figura 4.4.

En la gráfica de la figura 4.4, podemos apreciar como claramente contamos con un menor número de pasos respecto a lo que se aprecia en la gráfica de la figura 4.2. Debemos recordar que el modelo está programado para no detener la etapa de entrenamiento hasta alcanzada la condición de satisfacción de precisión del 95 %, de modo que al tener menos pasos, el entrenamiento resulta mucho más ágil, tanto en tiempo como en poder de cómputo.

Ahora bien, dejando de lado la eficiencia durante el entrenamiento, en la gráfica de la figura 4.4 podemos ver como la precisión no muestra un

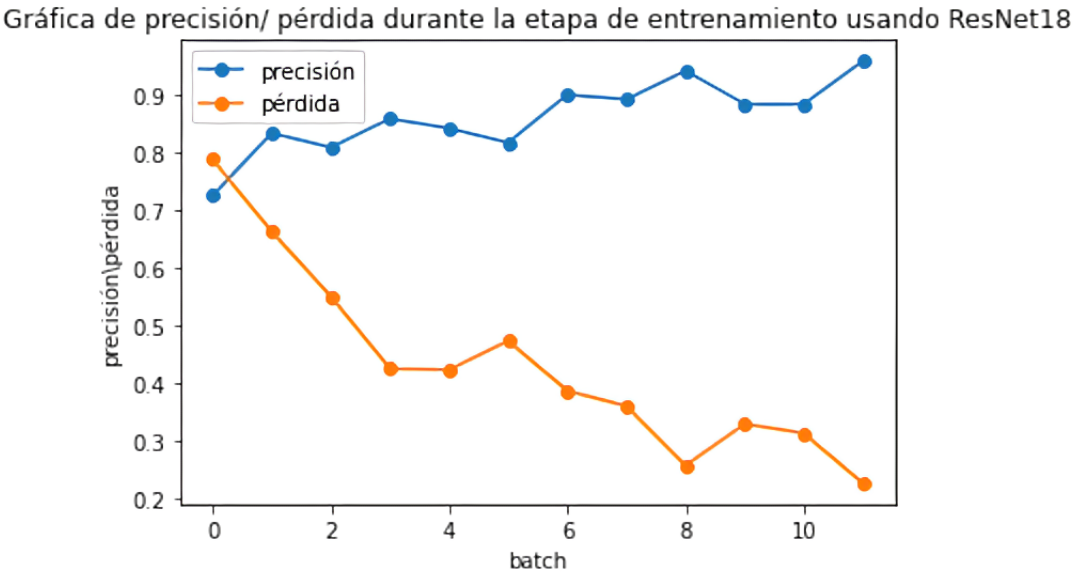


Figura 4.4: Gráfica de pérdida y precisión para durante el entrenamiento mediante la red ResNet 18

comportamiento evolutivo a lo largo del entrenamiento, teniendo un comportamiento cercano al de una recta. Como punto adicional, podemos observar saltos relevantes a lo largo de la curva, todo lo anterior nos habla de un aprendizaje poco representativo y al no mostrarse un comportamiento de estabilidad en los últimos pasos del entrenamiento, podemos entender que la condición de satisfacción se alcanzó de forma casi azarosa y nuevamente, entendemos que el aprendizaje resultó poco significativo y por lo tanto, poco notable.

Podemos observar un comportamiento análogo para la curva de pérdida de la gráfica de la figura 4.4, a diferencia de los resultados encontrados con la red VGG16, las curvas de pérdida y precisión no son complementarias. Ya mencionamos el acelerado y poco contundente crecimiento mostrado

en la curva de precisión, en el caso de la curva de pérdida, podemos ver como en las primeras etapas del entrenamiento la curva muestra una pendiente constante pero a partir del tercer lote de imágenes se aprecia un comportamiento un tanto caótico. La curva de pérdidas se estabiliza momentáneamente y posteriormente comienza a tener saltos que denotan el retroceso que también se mostró en la curva de precisión. Aunque el porcentaje de pérdidas final es pequeño, el comportamiento de la curva durante el proceso de entrenamiento denota un mal desempeño del modelo al entrenar con esta red neuronal.

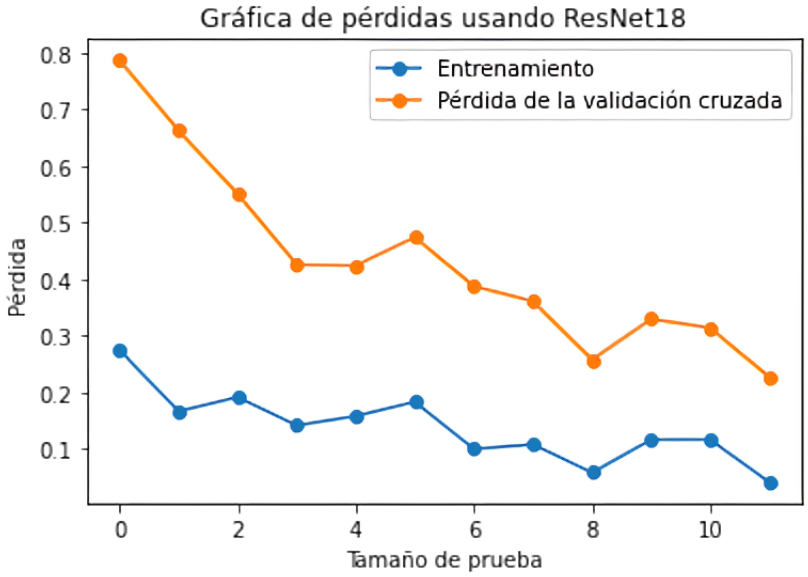


Figura 4.5: Gráfica de pérdidas con validación cruzada para ResNet18

En la gráfica de la figura 4.5 se encuentran las gráficas de las pérdidas tanto de la etapa de entrenamiento como del proceso de validación cruzada, dónde recordemos, se tomó una fracción de las mismas imágenes usadas para el entrenamiento con el objetivo de realizar pruebas aún durante el entrenamiento. Ya se ha comentado el rendimiento denotado en la curva de pérdida que es consistente con la evolución mostrada en la curva de precisión, podemos observar el mismo rendimiento en la gráfica de pérdidas. Primeramente, además de lo que ya se ha comentado, hay que destacar la gran brecha que existe entre ambas curvas de pérdida. La brecha observada alcanza una distancia de 50 % en promedio. Recordemos que la brecha existente entre las curvas de pérdida se conoce como brecha de generalización y el que sea grande hace referencia a un conjunto de datos poco representativo.

Cuando decimos que un conjunto de datos es poco representativo o no representativo nos referimos a que dicho conjunto de datos no puede capturar las características estadísticas en relación con otro conjunto de datos extraído del mismo dominio, como sucede con un conjunto de datos de entrenamiento y el conjunto de datos usado para la validación cruzada. Dado que en las curvas de pérdida podemos observar, por así decirlo, una mejora conforme nos movemos a lo largo del eje de las ordenadas pero aún así encontramos una gran brecha entre ambas curvas; diremos que tenemos un conjunto de datos de validación poco representativo.

Un conjunto de datos no representativo de entrenamiento significa que

el conjunto de datos de entrenamiento no proporciona información suficiente para aprender el problema, relativo al conjunto de datos de validación utilizado para evaluarlo. Podemos encontrar casos como este cuando el conjunto de datos de entrenamiento tiene muy pocos ejemplos en comparación con el conjunto de datos de validación. Esto indica que el modelo es capaz de aprender más con algunas posibles mejoras adicionales y que el proceso de capacitación se detuvo prematuramente. A pesar de lo que se puede pensar, se realizaron varias corridas de entrenamiento ya que debemos recordar, los grupos de prueba y entrenamiento se seleccionan al azar para tener condiciones parejas entre las evaluaciones del modelo con distintas redes neuronales. Del mismo modo, no se realizó una modificación de la condición de satisfacción con el fin de realizar una evaluación objetiva del modelo. Valdría la pena realizar las mismas pruebas con un conjunto de datos más equilibrado en lo que refiere a las clasificaciones, sin embargo, esto es cuestión de tiempo. En la actualidad, las bases de datos de imágenes radiológicas de pacientes COVID no son tan bastas como aquellas pertenecientes a pacientes con otras patologías.

Por otra parte, en la tabla 4.4 encontraremos la matriz de confusión obtenida para la etapa de pruebas del modelo usando la red ResNet18, notaremos la clara tendencia hacia los positivos verdaderos, tan sólo observando la diagonal de la matriz. Dados los resultados que obtuvimos con las

Tabla 4.4: Matriz de confusión para el modelo entrenado mediante la red neuronal ResNet18

Matriz de confusión para ResNet 18					
		Valores predichos por clase			
		COVID	Normal	Neumonía Viral	Opacidad Pulmonar
Valores reales	COVID	18	3	0	0
	Normal	0	27	0	1
	Neumonía Viral	0	0	25	0
	Opacidad Pulmonar	0	2	0	24

gráficas de las figuras 4.4 y 4.5, podemos asumir que nuestro clasificador está teniendo un mejor desempeño con este grupo de prueba, sin embargo, esto se desmentirá o comprobará mediante la curva ROC. Por lo pronto, es de destacar el buen desempeño del clasificador.

En la tabla 4.4 se encuentran los resultados obtenidos de acuerdo con las ecuaciones de la tabla 3.1 y los valores obtenidos en 4.4. Es notorio como las clasificaciones del modelo recaen principalmente en casos TP y TN sobre los FN y FP, es decir que la mayoría de las imágenes de prueba fueron correctamente clasificadas. Como resultado, tenemos valores muy altos de sensibilidad, precisión y especificidad.

Particularmente, la categoría COVID presentó métricas muy buenas, se encontraron precisión y especificidad del 100 %, así como una sensibilidad del 85.7 %. Lo anterior indica que, en lo que corresponde con esta categoría de clasificación y las imágenes del grupo de prueba, el modelo presenta una

nula dispersión y que, cuando se introduce una imagen que no pertenece a la categoría COVID, el modelo siempre distingue la imagen y la manda a otra clasificación. Además, la puntuación de sensibilidad también indica que el modelo puede identificar correctamente en un 85.7 % de las veces un estudio radiológico de un paciente COVID. En lo que refiere a la categoría Normal, la precisión presentada es de 84.4 % que aunque no es tan alta como en ejemplos anteriores, esta misma categoría presenta una sensibilidad del 96.4 % que resulta complementario con los resultados para la categoría COVID. Al tener un buen desempeño del modelo en identificar imágenes que no pertenecen a la categoría COVID y además una correcta estimación en otras categorías nos habla de cuán confiable es el modelo para predecir casos COVID, el cual es el objetivo principal de este trabajo.

Podemos encontrar que en general, todas las categorías de clasificación evaluadas para el modelo con esta red neuronal presentan buenos resultados y aunque en algunas de las categorías, los resultados de ciertas métricas son más débiles que en otras clasificaciones, los resultados parecen complementarse, tal como se comentó en el párrafo anterior. Y es que las categorías Neumonía Pulmonar y Opacidad Pulmonar resultaron tener métricas de evaluación muy sólidas acercándose mucho al 100 % o incluso llegando a la puntuación perfecta.

En este punto es importante subrayar dos hechos relevantes, en primer lugar debemos recordar los resultados obtenidos en las gráficas de las figuras 4.5 y 4.4. Recordemos que en párrafos atrás hablamos del mal ajuste

Tabla 4.5: Tabla resumen para el modelo entrenado mediante la red neuronal ResNet18.

	TP	FN	Total por clase	FP	TN
COVID	18	3	21	0	79
Normal	27	1	28	5	67
Neumonía Viral	25	0	25	0	75
Opacidad Pulmonar	24	2	26	1	73
	Precisión	Sensibilidad	Especificidad	Rango de error	
COVID	1	0.857142857	1	0.03	
Normal	0.84375	0.964285714	0.930555556	0.06	
Neumonía Viral	1	1	1	0	
Opacidad Pulmonar	0.96	0.923076923	0.986486486	0.03	

del modelo, por lo que aunque en este grupo de prueba se pudieron haber tenido resultados alentadores, no implica que sea la generalidad. Es importante mencionar que sí se realizaron pruebas con más imágenes y resultaron similares, pero, como ya se dijo, no podemos asegurar que siempre ocurra esto. Ahora bien, ya también se había mencionado que el modelo entrenado con ResNet18 no tiene un buen ajuste y que además, por la forma de las curvas de pérdida y precisión, el entrenamiento se detuvo prematuramente. De aquí que al evaluar los resultados de la tabla 4.5, se nota una sutil tendencia de incremento en el rango de error relacionada con un mayor número de ejemplares totales por clase.

Debemos notar una principal diferencia entre la ResNet18 y la VGG16,

de acuerdo con sus artículos [41] y [33] respectivamente, ResNet18 tiene menos filtros y complejidad respecto a la red VGG16, lo que aclara los resultados obtenidos para estas dos redes. A pesar de que aparentemente, ResNet18 es más profunda en relación con VGG16, la arquitectura de la segunda la hace más eficiente.

4.3. ResNet50

Como ya mencioné en varias secciones anteriores, la motivación para crear la estructura base de las redes neuronales ResNets implica conexiones de acceso directo para obtener la llamada red residual y poder agregar más capas evitando el problema de retropropagación.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figura 4.6: Tabla con recopilación de las arquitecturas de las redes ResNet18, ResNet34, ResNet50, ResNetXt101 y ResNetXt152. Figura obtenida de Wu, Huiyan & Xin, Ming & Fang, Wen & Hu, Hai-Miao & Hu, Zihao. (2019). Multi-Level Feature Network With Multi-Loss for Person Re-Identification. IEEE Access. PP. 1-1. 10.1109/ACCESS.2019.2927052. [6]

En la sección anterior conjuntamos el modelo propuesto con la red ResNet18, al igual que con la red neuronal anterior, ResNet sigue dos reglas de diseño simples. Las capas tienen la misma cantidad de filtros para el mismo tamaño de mapa de características de salida, pero la cantidad de filtros se

duplicó en caso de que el tamaño del mapa de características se redujera a la mitad para preservar la complejidad del tiempo por capa. Además, las conexiones de acceso directo se agregaron a esta red simple. Si bien las dimensiones de entrada y salida eran las mismas, los atajos de identidad se usaron directamente. Con un aumento en las dimensiones, había dos opciones a considerar. La primera fue que el atajo aún realizaría el mapeo de identidad mientras que las entradas cero adicionales se rellenarían para aumentar las dimensiones. La otra opción era usar el atajo de proyección para hacer coincidir las dimensiones.

En la tabla de la figura 4.6 encontramos un resumen general de la estructura de varias redes ResNets. Podemos ver como la arquitectura de la ResNet50 tiene una particularidad en su diseño que la distingue de la ResNet34, su predecesora, y de otras redes en general. Y es que el bloque de construcción se modificó en un diseño de cuello de botella como respuesta a las preocupaciones sobre el tiempo de entrenamiento. Por lo tanto, cada uno de los bloques de 2 capas en Resnet34 se reemplazó con un bloque de cuello de botella de 3 capas, formando la arquitectura Resnet50. Los resultados obtenidos por He, K. et al. [41], ResNet50 presenta una precisión mucho más alta que ResNet18, ResNet34 y VGG16. Por su arquitectura novedosa es que la ResNet50 obtuvo el primer lugar en detección de ImageNet en 2015 y es una red obligada al evaluar un modelo con distintas redes neuronales.

En la figura 4.7 tenemos la gráfica de pérdida y precisión obtenidas

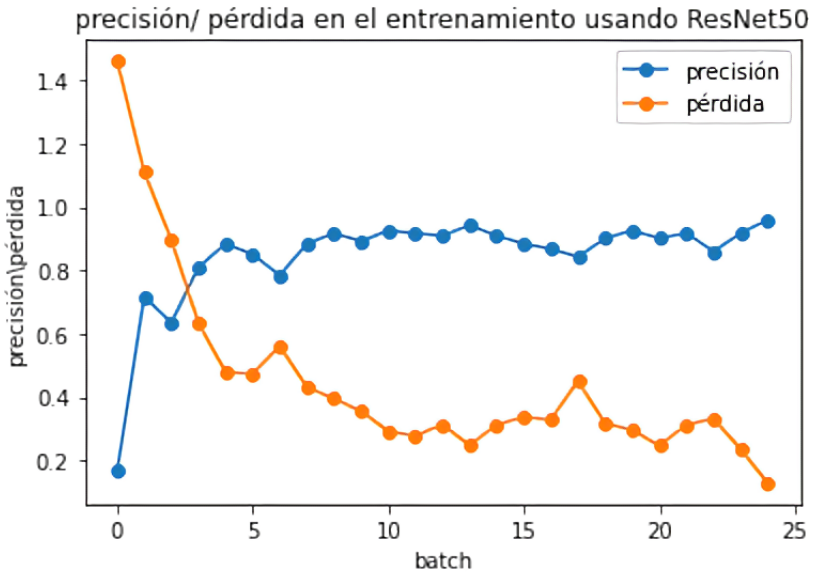


Figura 4.7: Gráfica de pérdida y precisión para durante el entrenamiento mediante la red ResNet 50

durante la etapa de entrenamiento del modelo usando ResNet50 y la optimización AdamNet. A diferencia del caso anterior, sí podemos encontrar una complementación entre ambas curvas. Podemos observar que incluso algunas cúspides de la curva de pérdidas se reflejan en valles en la curva de precisión.

Ahora, la gráfica de precisión tiene un buen desarrollo, de acuerdo a lo que podemos observar en la gráfica de la figura 4.7. Podemos apreciar algunos saltos y retrocesos pero en general se puede distinguir su evolución a lo largo del entrenamiento. Notamos como conforme crece el número de pasos, la curva alcanza cierta estabilidad y aunque al principio se notan saltos abruptos, estos se van mermando conforme se aproxima al final del proceso de entrenamiento.

No podemos decir lo mismo de la curva de pérdidas. Aunque a grandes rasgos encontramos la forma de una curva exponencial negativa, que es el caso deseable, debemos destacar la abrupta caída de la pérdida al final del entrenamiento. A diferencia de lo que encontramos con la precisión, la gráfica de pérdidas no muestra estabilidad en ningún punto del entrenamiento. Sin embargo, estos picos son esperables dado que usamos el descenso de gradiente durante la construcción del modelo propuesto y además tenemos como recurso el optimizador AdamNet, los cuales son procesos estocásticos.

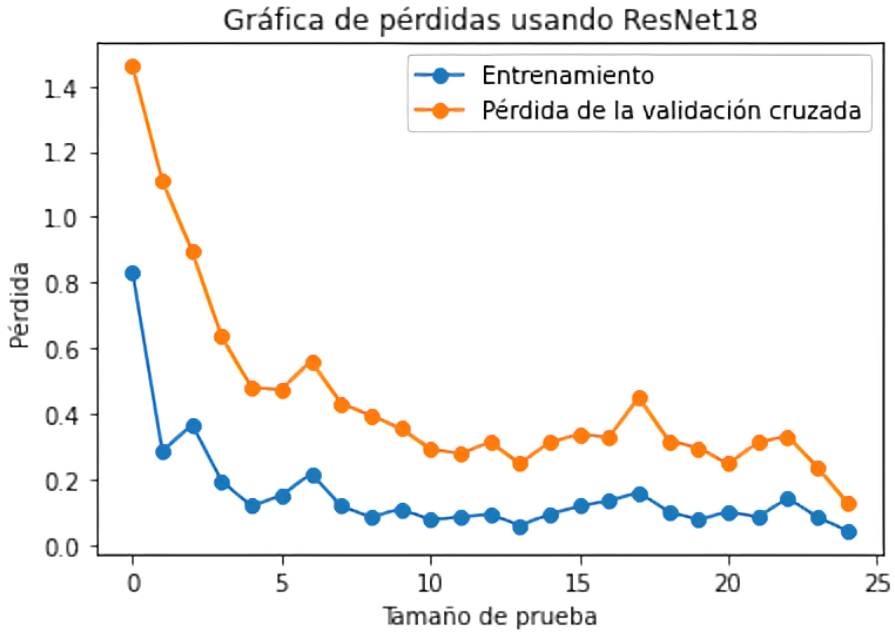


Figura 4.8: Gráfica de pérdidas con validación cruzada para ResNet50

La gráfica de pérdida para la etapa de entrenamiento y validación cruzada se encuentra en la figura 4.8. Por la pequeña brecha y la discusión del párrafo anterior, podemos decir que el modelo presenta un buen ajuste y

por lo tanto, una buena generalización cuando se entrena con ResNet50.

Así mismo, debemos considerar el número de pasos necesarios para llegar a la condición de satisfacción. En lo que respecta a las 2 redes anteriores, VGG16 utilizó una cantidad similar a ResNet50; mientras que a la red ResNet18 le tomó muchos menos pasos. En la sección anterior ya discutimos el hecho de que aunque la ResNet18 tuvo un entrenamiento mucho más eficaz, la forma de sus curvas de pérdida indica que el entrenamiento se detuvo prematuramente y que no tiene buena generalización. Ahora, aunque las redes VGG16 y ResNet50 parecen tener un rendimiento similar en lo que corresponde al entrenamiento del modelo, resulta importante mencionar el tiempo total de cómputo. Mientras que a la red VGG16 le tomó 162 minutos completar el entrenamiento hasta la condición de satisfacción, a la red ResNet50 le tomó apenas 54 minutos. Por lo que el entrenamiento resulta más eficiente al utilizar la red ResNet50.

Tabla 4.6: Matriz de confusión para el modelo entrenado mediante la red neuronal ResNet50.

Matriz de confusión para ResNet 50					
		Valores predichos por clase			
		COVID	Normal	Neumonía Viral	Opacidad Pulmonar
Valores reales	COVID	23	2	0	0
	Normal	0	23	2	4
	Neumonía Viral	0	0	24	0
	Opacidad Pulmonar	0	3	0	19

En la tabla 4.6 se encuentra la matriz de confusión para el modelo entrenado con ResNet50 y la optimización AdamNet. Encontramos que tal como sucedió con ResNet18, la mayoría de las clasificaciones realizadas por el modelo resultaron acertadas. Sin embargo, recordemos que en este caso sí se encontraron pruebas de una buena generalización. De los resultados encontrados en la tabla 4.6 y las ecuaciones de la tabla 3.1 en la metodología, se hallaron los valores mostrados en la tabla 4.7.

Tabla 4.7: Tabla resumen de resultados para ResNet 50, de acuerdo con las expresiones contenidas en la tabla 3.1

	TP	FN	Total por clase	FP	TN
COVID	23	2	25	0	75
Normal	23	6	29	5	66
Neumonía Viral	24	0	24	2	74
Opacidad Pulmonar	19	3	22	4	74
	Precisión	Sensibillidad	Especificidad	Rango de error	
COVID	1	0.92	1	0.02	
Normal	0.821428571	0.793103448	0.929577465	0.11	
Neumonía Viral	0.923076923	1	0.973684211	0.02	
Opacidad Pulmonar	0.826086957	0.863636364	0.948717949	0.07	

Para la categoría COVID, tenemos que de las 100 imágenes en el conjunto de prueba, 21 estaban etiquetadas como pertenecientes a clase COVID. Encontramos una clasificación alentadora en lo que corresponde a las columnas de la parte superior de la tabla 4.7, tal que las clasificaciones

positivas verdaderas (TP) y negativas verdaderas (TN) fueron mayoritarias en contraste con las falsas. Lo anterior se refleja en los resultados de precisión, especificidad, sensibilidad y rango de error.

De los resultados de la tabla 4.7 para la clase COVID, tenemos una precisión del 100 %. Recordando que la precisión es la taza de predicciones positivas correctas sobre el total de predicciones hechas por categoría, una precisión tan alta indica que el modelo presenta una nula dispersión en lo que refiere al grupo de prueba utilizado. Además, como los resultados de la gráfica de la figura 4.8 indican una buena generalización, así como un buen ajuste del modelo, podemos decir que además, en general, el modelo presenta una baja dispersión.

Ahora bien, en cuanto a la sensibilidad en la clasificación COVID, tenemos un total de 92.0 %. Es el valor más alto para sensibilidad que hemos encontrado dentro de los resultados obtenidos para nuestro modelo con las redes evaluadas en la categoría COVID. Así pues, de nuestro grupo de prueba, el 92 % de los casos positivos para COVID fueron correctamente identificados por nuestro algoritmo. Lo que indica una alta fiabilidad.

Finalmente y tal como pasó con la precisión, tenemos una especificidad del 100 % por lo que, para las imágenes en el grupo de prueba, todos los casos que estaban etiquetados con alguna de las otras 3 categorías fueron descartados correctamente. Este resultado confirma la alta fiabilidad mostrada con la sensibilidad. Hasta el momento podemos decir que, al menos

en el contexto de la clasificación de casos COVID, el modelo propuesto entrenado con la red ResNet50 y la optimización AdamNet muestra los mejores resultados.

De acuerdo con Wong, H. y colaboradores [16], la sensibilidad del diagnóstico de la neumonía producida por el virus Sars-CoV2 mediante radiografías a través del ojo entrenado de un radiólogo es de aproximadamente 69 %. Además, de acuerdo con Gálvez & Montoya en [42] , en general, los errores de un especialista varían entre un 2 a un 20 por ciento. Y tal como se hace para evaluar al modelo propuesto, debemos considerear posibles errores diagnósticos. Principalmente, los diagnósticos omitidos tanto como los diagnósticos incorrectos o tardíos, que de acuerdo con Galvez y Montoya [42] van de $10 - 15\,\%$. Así mismo, los autores indican falsos negativos estimadoes en un 25 % y los falsos positivos en un 5 %. Es claro que estos errores varían dependiendo de distintos factores y es importante destacar que al evaluar la habilidad de clasificación de la mente humana, se deben considerar varios tipos de error. Por lo que nuestros resultados para esta categoría son equiparables con los de un especialista según Wong, H. [16] y por lo tanto, el modelo propuesto entrenado mediante ResNet50 podría ser de gran utilidad como herramienta de diagnóstico o como medio didáctico para futuros médicos radiólogos.

Ahora, en lo que refiere a las demás categorías, para el caso de la Neumonía Viral, encontramos que sus evaluaciones se encuentran sobre el 92 % en todos los casos. Particularmente, encontramos que el modelo es capaz

de identificar correctamente tanto los casos positivos verdaderos como los negativos verdaderos en lo que corresponde a esta categoría. Esto es importante porque, también hemos visto que, de acuerdo con H. S. Maghdid y colaboradores [9], una de las principales limitantes del diagnóstico clínico de pacientes COVID mediante estudios radiológicos es la similitud que tiene en el sentido radiológico con la Neumonía Viral.

Del mismo modo, en el caso de la opacidad pulmonar, encontramos resultados mayores al 82 % en las tres métricas, además de un rango de error del 7 %. Aunque no alcanzan los porcentajes de otras categorías, aún se encuentran en un rango aceptable.

Ahora, en la sección 1 mencionábamos como los modelos *Computer Aided Designs* CAD pos sus siglas en inglés, han demostrado facilitar la labor en el área médica, principalmente en la detección de Cáncer de Mama, nódulos pulmonares, mamogramas, entre otros. Particularmente, en el campo de detección del COVID-19 y Neumonía Viral mediante modelos CAD, se han encontrado resultados de diagnóstico con sensibilidades de hasta 83.61 %, 96.8 % y 80.02 % en el 2020 [12], [17], [18] y [9], resultados que son comparables con el desempeño de un especialista e incluso con los resultados obtenidos por la prueba RT-PCR. Así, encontramos que los resultados obtenidos para el modelo propuesto resultan equiparables con otros en su categoría.

Finalmente, observamos que la categoría Normal es la clasificación con

un mayor rango de error, 11 %. Con una puntuación de 79.3 %, la sensibilidad resulta la métrica más débil para esta categoría, seguida por la precisión que alcanza un porcentaje de 82.1 %. Por lo que en la categoría Normal encontramos dispersión y errores al clasificar imágenes positivas en esta categoría. Sin embargo, se calcula una especificidad del 93.0 %, lo que indica que aunque el modelo podría mal clasificar un paciente sano con alguna de las otras patologías, tiene un bajo índice de error en mal clasificar pacientes enfermos en la categoría de pacientes saludables. Esto es relevante para fines clínicos pues significa una disminución en la carga de trabajo para un especialista.

4.4. ResNetxt101

La red ResNetxt es una colaboración entre UC de San Diego y Facebook AI Research (FAIR por sus siglas en inglés). La red se creó con el objetivo de mejorar la clasificación de objetos en imágenes. El nombre del modelo significa *la siguiente dimensión* y hace referencia al siguiente paso en lo que refiere a la dimensión de cardinalidad. Al momento en que se escribe la presente tesis, *ResNetXt*, tiene uno de las menores tasas de error en la categoría de detección de objetos con sólo un rango de error del 3.03 % [43].

Ya hemos comentado como ResNet surge con la idea de conexiones residuales en respuesta a solucionar problemas de saturación en la exactitud y degradación en redes profundas. También hemos hablado ya de algunas de las variantes de las redes ResNet, tal como la ResNet50 y la ResNet18. El marco de aprendizaje residual utilizado en la arquitectura ResNetXt101 facilita el entrenamiento de redes aún más profundas y reformula las capas para aprender funciones residuales con referencia a las entradas de la capa. Esto hace que el modelo ResNetXt101 sea más fácil de converger y permite que gane precisión a partir de una profundidad considerablemente mayor.

ResNetXt combina características de sus antecesores; *ResNet*, *Inception* y la convolución agrupada de la *AlexNet*. Además de que da un salto pues en lugar de ser un modelo tipo Red-en-Red, *ResNetXt* es Red-en-neurona, es decir, en vez de aplicar una función lineal en una sola neurona simple que se multiplica x_i veces en cada ruta, encontraremos una función no lineal

para cada una de las rutas.

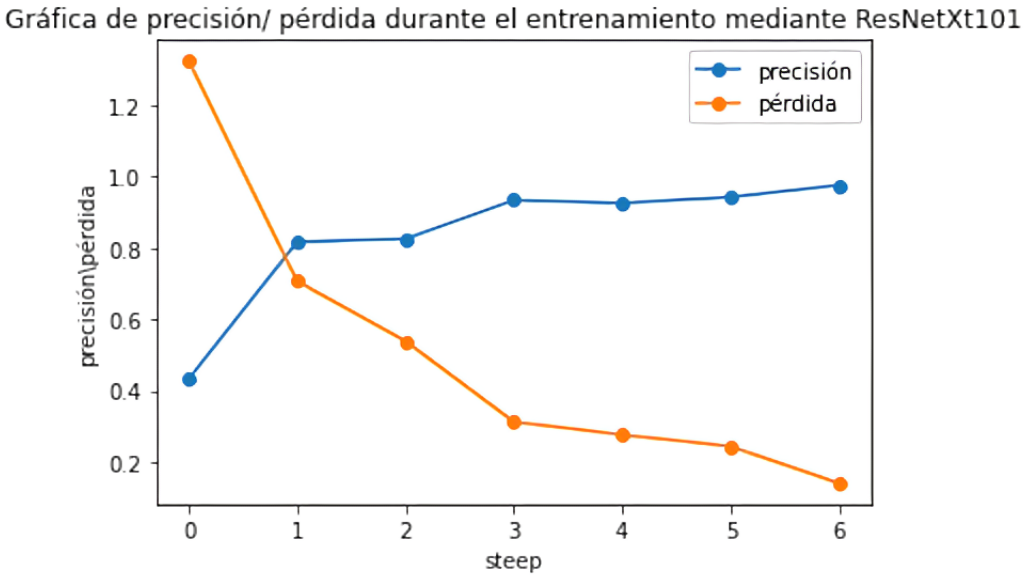


Figura 4.9: Gráfica de pérdida y precisión para el modelo en la etapa de entrenamiento realizado con la red ResNet Xt 101

En 4.9 se encuentra la gráfica de pérdida y precisión durante el entrenamiento realizado con la red ResNet Xt101. Se puede apreciar que efectivamente se sigue una tendencia positiva para la curva de precisión conforme avanza el entrenamiento, mostrando que a mayor número de pasos, mayor precisión alcanzamos. El comportamiento anterior resulta consistente al compararse con la tendencia de la curva de pérdidas, la cuál es propensa a disminuir con la evolución de los pasos. Es decir que ambas curvas se complementan y muestran un progreso adecuado. Notamos además que el entrenamiento fue muy eficiente, tomando apenas 6 pasos para llegar a la condición de satisfacción. Esto es de gran importancia debido a la implicación con los recursos computacionales, a pesar de ser una red bastante amplia, es muy amigable con la memoria RAM.

Ahora, aunque podríamos decir que las gráficas tienen un buen desarrollo, tal que tenemos curvas suaves sin saltos ni retrocesos, es necesario destacar la forma de la curva de precisión. Notamos, tal como se muestra en la gráfica de la figura 4.9, la precisión presenta un salto abrupto de casi un 40 % de incremento entre un paso y otro y luego se mantiene más o menos constante. Lo anterior denota un aprendizaje poco significativo.

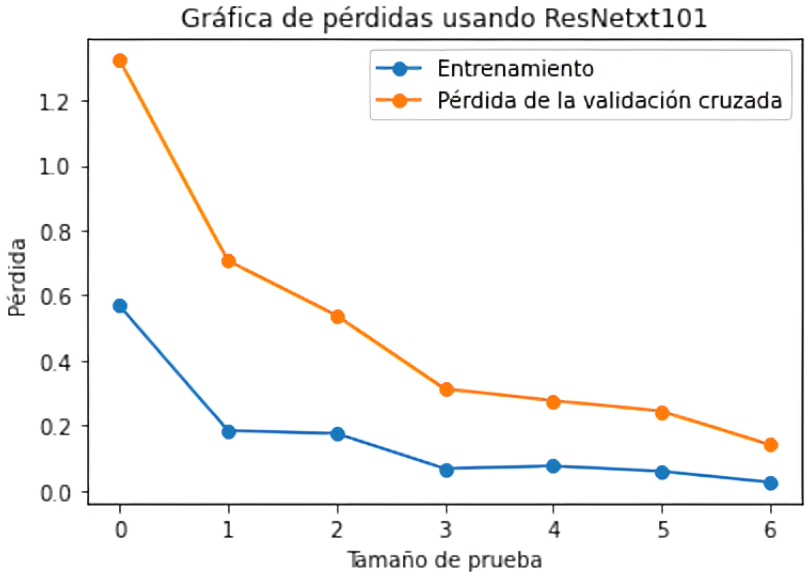


Figura 4.10: Gráfica de pérdidas con validación cruzada para ResNet Xt 101

En lo que refiere a las curvas de pérdida del modelo en las etapas de entrenamiento y validación cruzada mostradas en la gráfica de la figura 4.10, tenemos que ambos casos presentan una evolución consistente con lo discutido para la precisión. Además, resulta importante mencionar la brecha entre las curvas de entrenamiento y validación cruzada de la figura 4.10, tal que las pérdidas encontradas durante la etapa de validación cruzada parecen mucho menores a las encontradas en la etapa de entrenamiento.

Por lo tanto, tenemos un modelo con un subre ajuste y mala generalización de los resultados.

Tabla 4.8: Matriz de confusión para el modelo entrenado mediante la red neuronal ResNet Xt 101.

Matriz de confusión para ResNet Xt 101					
		Valores predichos por clase			
		COVID	Normal	Neumonía Viral	Opacidad Pulmonar
Valores reales	COVID	15	3	0	0
	Normal	0	18	3	7
	Neumonía Viral	0	0	27	0
	Opacidad Pulmonar	0	0	0	27

Para un grupo de prueba se obtuvo la matriz de confusión que se encuentra en la tabla 4.8. Aunque en la mayoría de los casos, el modelo clasificó correctamente las imágenes en su respectiva categoría, de los resultados encontrados en la tabla resumen 4.9 podemos afirmar que esta es la red neuronal con rangos de error más altos, alcanzando el 13 %. La clase Normal fue la categoría más castigada, presentando una sensibilidad, precisión y especificidad de 64.3 %, 85.7 % y 95.8 % respectivamente. Lo anterior resulta especialmente preocupante en este caso ya que aunque la clase COVID presenta una sensibilidad del 83.4 % y una especificidad del 100 %, el modelo no está distinguiendo correctamente los casos que no están saludables y adjudica esta etiqueta a radiografías de pacientes con

Opacidad Pulmonar y Neumonía Viral.

Tabla 4.9: Resumen de los resultados obtenidos del modelo propuesto entrenado con ResNet Xt101 de acuerdo con las ecuaciones contenidas en la tabla 3.1

	TP	FN	Total por clase	FP	TN
COVID	15	3	18	0	82
Normal	18	10	28	3	69
Neumonía Viral	27	0	27	3	70
Opacidad Pulmonar	27	0	27	7	66
	Precisión	Sensibilidad	Especificidad	Rango de error	
COVID	1	0.8333	1	0.03	
Normal	0.8571	0.6429	0.9583	0.13	
Neumonía Viral	0.9	1	0.9589	0.03	
Opacidad Pulmonar	0.7941	1	0.9041	0.07	

4.5. Inception V3

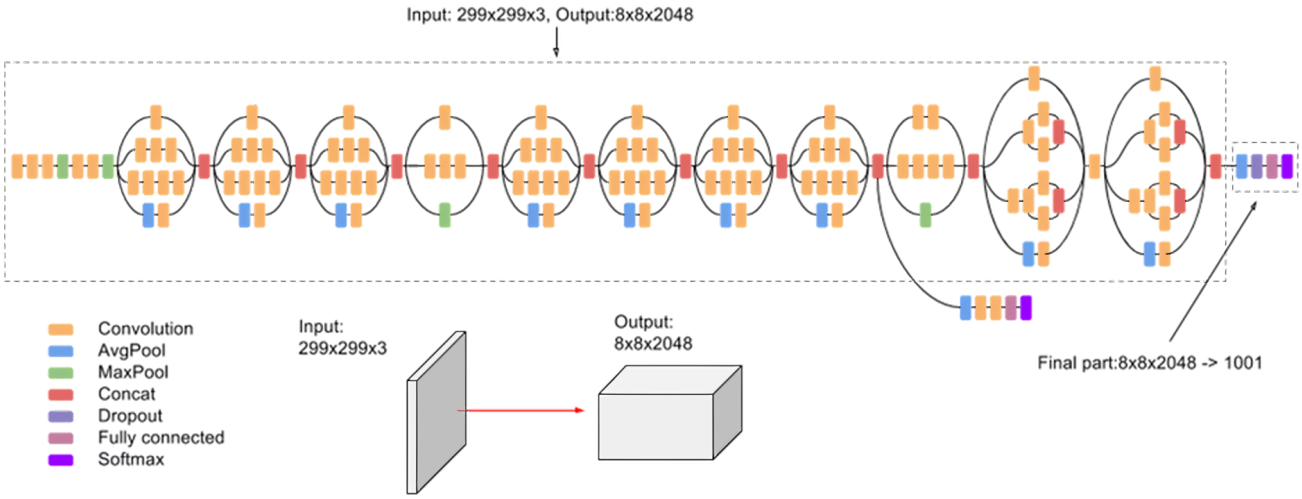


Figura 4.11: Arquitectura de la red Inception V3. Figura obtenida de [7]

Inception V3 es un modelo de reconocimiento de imágenes que se desarrolló en el 2015 a partir de su predecesora, la red Inception V1. Como dato curioso, la red obtiene su nombre de la película de 2010 dirigida por Christopher Nolan, ya que su arquitectura se basa en la construcción de redes en módulos en lugar de apilar capas, tal como sucede en la trama de la película. La arquitectura de Incepton V3 posee 24M parámetros. La red se basa en la red Inception V1, rescatando conceptos de la red Inception V2, tratando de evitar los cuellos de botella representacionales. Por lo que se controlan los tamaños de entrada de un bloque a otro, obteniendo cálculos más eficientes basados en métodos de factorización.

En la figura 2.5 se encuentra un gráfico que ilustra la arquitectura de la red Inception V3, se aprecia que el modelo está formado por bloques

de compilación simétricos y asimétricos, que incluyen convoluciones, reducción promedio, agrupación máxima, concatenaciones, retirados y capas completamente conectadas. La normalización por lotes se usa ampliamente en todo el modelo y se aplica a las entradas de activación. La pérdida se calcula usando Softmax.

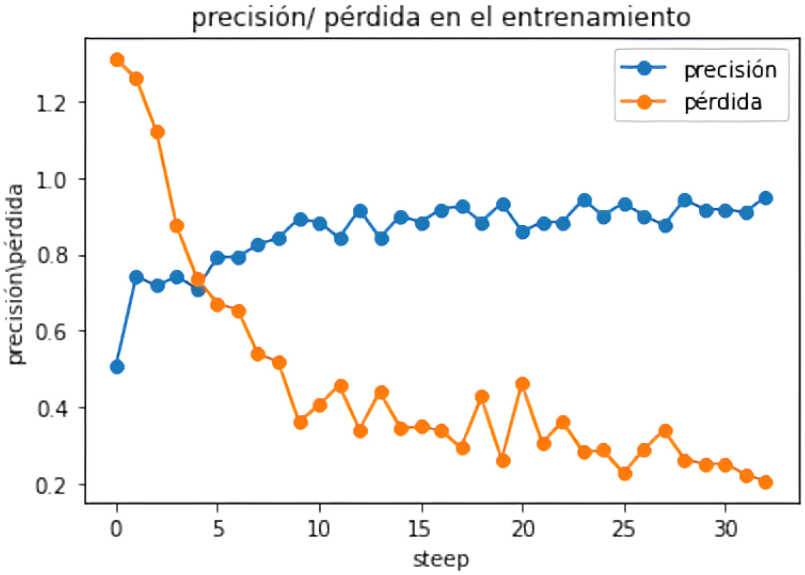


Figura 4.12: Gráfica de pérdida y precisión para durante el entrenamiento mediante la red Inception V3

La implementación actual de Inception v3 se encuentra en el perímetro de la vinculación de entrada. Las imágenes se recuperan del sistema de archivos, se decodifican y, luego, se procesan. Hay diferentes tipos de etapas de procesamiento previo disponibles, desde moderadas hasta complejas. Si usamos las etapas de procesamiento previo más complejas, la canalización de entrenamiento se vinculará al procesamiento previo.

A diferencia de las redes que hemos evaluado hasta el momento, la red Inception V3 fue construida con el objetivo de eficientar los recursos computacionales sin castigar la profundidad de análisis. Además, en comparación con las redes tipo VGG, las redes Inception han probado ser computacional-mente más eficientes, tanto en términos de la cantidad de parámetros generados por la red como en el costo económico incurrido (memoria y otros recursos). Si se van a realizar cambios en una red de inicio, se debe tener cuidado para asegurarse de que no se pierdan las ventajas computacionales. Así, la adaptación de una red Inception para diferentes casos de uso resulta ser un problema debido a la incertidumbre sobre la eficiencia de la nueva red. En un modelo de Inception v3, se sugirieron varias técnicas para optimizar la red para aflojar las restricciones y facilitar la adaptación del modelo. Las técnicas incluyen convoluciones factorizadas, regularización, reducción de dimensiones y cálculos en paralelo. En nuestro caso, tal como se comentó y mencionó con las redes anteriores, la red se importó pre-entrenada con ImageNet y sólo se modificó la cantidad de características de salida para que se consideraran nuestras cuatro clases y además se utilizó el algoritmo de optimización Adam.

En la gráfica de la figura 4.12 se observan las curvas de pérdida y precisión encontradas durante el entrenamiento del modelo propuesto mediante la red Inception V3 con la optimización Adam. Lo primero que salta a la vista es la pausada evolución de las curvas, a pesar de la arquitectura discutida en párrafos anteriores, el proceso de entrenamiento tomó más pasos

que en caos anteriores. Notamos que existen muchos saltos en la evolución de las curvas, debemos recordar que esto se debe principalmente al proceso estocástico seguido en la construcción del modelo. Dejando de lado lo mencionado anteriormente, aún se deben resaltar los saltos a la mitad del entrenamiento, ya que sí denotan un importante retroceso en la evolución, lo que posiblemente causó la cantidad de pasos necesarios para llegar a la condición de satisfacción. Sin embargo, sí se nota cierta estabilidad en la conclusión del entrenamiento.

Ya se ha comentado la importancia de tener un entrenamiento eficaz que use pocos recursos computacionales. Dado que Inception V3 se construyó con esa motivación, aunque fue un proceso con muchos pasos, el proceso fue ágil y económico en términos de memoria RAM, tomando sólo 45 minutos y la mitad de la memoria disponible para llegar al punto de término. Por lo que podemos decir que el modelo se complementa bien con la condiciones fijadas.

Ahora, en la gráfica de la figura 4.13, se encuentra la gráfica de pérdidas encontradas durante el entrenamiento y validación cruzada. Ambas curvas muestran una evolución negativa correlacionada con el incremento de pasos, además de alcanzar la estabilidad hacia el final de la gráfica. Sin embargo, encontramos una brecha de magnitud considerable entre ambas gráficas. Sabemos que es esperable que la gráfica de entrenamiento se encuentre por debajo de la correspondiente a la validación cruzada. En este caso, la notable brecha indica que fue mucho más fácil predecir correctamente los

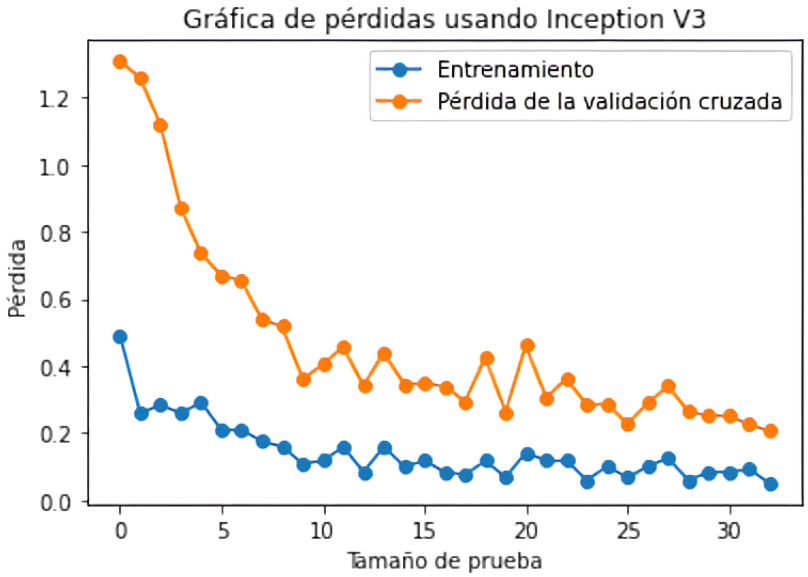


Figura 4.13: Gráfica de pérdidas con validación cruzada para Inception V3

ejemplares del grupo de validación cruzada en contraste con los del grupo de entrenamiento, lo que nos habla de un sobre ajuste y no nos permite generalizar con confianza los resultados obtenidos.

Tabla 4.10: Matriz de confusión para el modelo propuesto entrenado con la red Inception V3

Matriz de confusión para Inception V3					
		Valores predichos por clase			
		COVID	Normal	Neumonía Viral	Opacidad Pulmonar
	COVID	25	1	0	0
	Normal	0	21	0	0
	Neumonía Viral	0	0	31	0
	Opacidad Pulmonar	3	0	0	19

Tabla 4.11: Resumen de resultados para Inception V3 de acuerdo con las ecuaciones contenidas en la tabla 3.1

	TP	FN	Total por clase	FP	TN
COVID	25	1	26	3	71
Normal	21	0	21	1	78
Neumonía Viral	31	0	31	0	69
Opacidad Pulmonar	19	3	22	0	78
	Precisión	Sensibilidad	Especificidad	Rango de error	
COVID	0.892857	0.9615	0.959459459	0.04	
Normal	0.9545	1	0.9873	0.01	
Neumonía Viral	1	1	1	0	
Opacidad Pulmonar	1	0.863636	1	0.03	

En la tabla 4.10 se encuentra la matriz de confusión y con los valores obtenidos se calculan las métricas definidas en la tabla 3.1, los resultados se resumen en la tabla 4.11. De la matriz de confusión encontramos que la mayoría de los elementos del conjunto de prueba fueron clasificados correctamente como positivos verdaderos, teniendo sólo 4 clasificaciones falsas negativas. De este modo, los rangos de error en las 4 categorías no rebasan el 4 %.

Ahora bien, para la clasificación COVID se obtuvo una precisión del 89.3 %. Dada la ecuación para la precisión contenida en la tabla 3.1, la clase COVID obtiene el puntaje más bajo para esta métrica tal que de

los 4 errores que tuvo la prueba, 3 se clasificaron como COVID siendo que pertenecían a la categoría Opacidad Pulmonar. Además, una imagen etiquetada con COVID tuvo una predicción Normal, lo que también afectó las otras métricas, incluido el rango de error. Encontramos que el rango de error más alto pertenece a esta categoría.

Los casos comentados en el párrafo anterior también afectan las métricas de evaluación de las demás clasificaciones. No obstante, en todas las categorías encontramos sensibilidades mayores al 86.4 %, así como especificidades mayores al 96.0 %. Lo que indica que en general el modelo tiene una baja dispersión y una distinción confiable entre casos positivos verdaderos y negativos verdaderos para este grupo de prueba.

En la discusión realizada para las gráficas de las figura 4.12 y 4.13 se habló de un sobre ajuste y de una poco confiable generalización del modelo. De este modo, se explican los altos resultados obtenidos en las evaluaciones del grupo de prueba y aunque, basándonos únicamente en los resultados de la gráfica de la figura 4.13, no es posible generalizar, debemos considerar que las imágenes de la validación cruzada no son las mismas del grupo de prueba.

CAPÍTULO 5

CONCLUSIONES

Se propuso un modelo de inteligencia Artificial (IA), específicamente, con la metodología del Deep Learning. El modelo demostró aptitudes para la detección de diversas patologías pulmonares como la Neumonía Pulmonar, la Neumonía producida por el virus del COVID-19 o la Opacidad Pulmonar. La estructura del modelo se pensó de tal forma que se entrenara mediante diversas redes neuronales clasificadas como State of Art, dichas redes se encontraban pre entrenadas con la base de datos ImageNet. La base de datos usada para entrenar al modelo se encuentra disponible para el público en general en la plataforma Kaggle [8]. La base de datos se amplió tras la colaboración con un médico radiólogo del Hospital Médi-

ca Avanzada Contigo. El doctor nos suministró 64 imágenes de pacientes hospitalizados tras ser diagnosticados con COVID-19. Las imágenes fueron útiles para realizar pruebas del desempeño del modelo, basadas en su diagnóstico médico profesional.

El modelo fue probado con métricas de evaluación de clasificadores tipo límite, tras ser entrenado mediante 5 redes neuronales. Las redes neuronales elegidas fueron seleccionadas debido a características arquitectónicas de interés como el enfoque de red residual, el número de capas convolucionales, la estructura de cuello de botella, entre otras. Es importante destacar que dado el proceso de evaluación de los resultados, nos encontramos frente a métricas que entran dentro de la descripción del método Montecarlo. Debemos recordar que el Montecarlo, es un método no determinista o estadístico numérico, usado para aproximar expresiones matemáticas complejas y costosas de evaluar con exactitud. Por lo que los porcentajes presentados en la presente tésis son las aproximaciones a un resultado máximo a obtener en el caso de un experimento en particular.

Habiendo mencionado lo anterior, la red neuronal con la que el modelo tuvo resultados destacables fue ResNet50, alcanzando 95 % de exactitud global y exactitud mínima por clase de 89 %. El modelo mostró tener una buena evolución en el entrenamiento con un buen ajuste y generalización al realizarse pruebas con grupos de validación cruzada tras entrenarse con dicha red. Asimismo, en las etapas de prueba, se alcanzaron rangos de error máximo de 11 %, además de precisiones sensibilidades y especifici-

dades por categoría de al menos 79.3 % y hasta el 100 %. Los resultados encontrados con esta red nos dan indicios de la viabilidad de usar el modelo propuesto como una herramienta diagnóstica para un especialista. Sin embargo, es importante recordarle al lector las condiciones de evaluación de los métodos descritos en este trabajo, por lo que resultaría importante realizar pruebas más sofisticadas que evalúen las probabilidades individuales de cada predicción como pueden ser las curvas ROC o los mapas de calor. El trabajo realizado en la presente tesis es un buen precedente para un proyecto con evaluaciones más estrictas y que, por decir, se pudiera usar con tranquilidad como herramienta en el ámbito médico.

En lo que refiere a las otras redes neuronales, se encontró que el modelo presenta sobre ajuste evidente, cuando se entrena con redes más sofisticadas como ResNet Xt101 o Inception V3. Por su parte y aunque el modelo también mostró rangos de error de hasta 9 %, además de precisiones, sensibilidades y especificidades de al menos 78.3 % al combinarse con la red VGG16, se encontró que el entrenamiento con esta red resulta muy tardado y requiere de muchos recursos computacionales.

Recordando que en todos los experimentos se fijó una condición en la precisión de 95 % para detener el entrenamiento, en el caso de la red VGG16 tomó 25 pasos y al rededor de 3 horas para llegar a la condición de satisfacción fijada, a diferencia de otras redes con las cuales el entrenamiento fue mucho más ágil.

Podemos concluir que el modelo demuestra un mejor desempeño en la etapa de entrenamiento cuando se implementan redes neuronales convolucionales de pocas capas y su desempeño se ve particularmente favorecido cuando implementamos la red neuronal $ResNet50$. Lo cual no es de extrañar pues ResNet50 además de contar con bloques residuales, presenta la estructura de cuello de botella que permiten optimizar el tiempo de entrenamiento con aprendizajes significativos. Los resultados son consistentes con la teoría, tal que el Large Scale Visual Recognition Challenge (LSVRC) o Concurso de Reconocimiento Visual a Gran Escala reconoce a ResNet50 como ganadora del 2015 tras presentar una taza de error del 3.6 % en contraste con los errores presentados por Inception o VGG de 6.7 % y 7.3 %, respectivamente de acuerdo con Russakovsky, O. y colaboradores [31].

Adicionalmente, de acuerdo con Wong, H. y colaboradores [16], la sensibilidad del diagnóstico de la neumonía producida por el virus Sars-CoV2 mediante radiografías a través del ojo entrenado de un radiólogo es de aproximadamente 69 %. Por su parte, Gálvez & Montoya en [42] , indican que en general, los errores de un especialista varían entre un 2 a un 20 por ciento. Y tal como se hace para evaluar al modelo propuesto, debemos considerar posibles errores diagnósticos. Principalmente, los diagnósticos omitidos tanto como los diagnósticos incorrectos o tardíos, que de acuerdo con Galvez y Montoya [42] van de $10 - 15$ %. Así mismo, los autores indican falsos negativos estimados en un 25 % y los falsos positivos en un 5 %. Es claro que estos errores varían dependiendo de distintos factores y es importante

destacar que al evaluar la habilidad de clasificación de la mente humana, se deben considerar varios tipos de error. Por lo que nuestros resultados para esta categoría son equiparables con los de un especialista según Wong, H. [16] y por lo tanto, el modelo propuesto entrenado mediante ResNet50 podría ser de gran utilidad como herramienta de diagnóstico o como medio didáctico para futuros médicos radiólogos. Es mi labor recordarle al lector las debilidades de la metodología Montecarlo, sin embargo, los resultados obtenidos representan un buen precedente para nuestro modelo y permiten el continuar con pruebas más rigurosas a futuro.

Ahora, en lo que refiere a las demás categorías, para el caso de la Neumonía Viral, encontramos que sus evaluaciones se encuentran sobre el 92 % en todos los casos. Particularmente, encontramos que el modelo es capaz de identificar correctamente tanto los casos positivos verdaderos como los negativos verdaderos en lo que corresponde a esta categoría. Esto es importante porque, también hemos visto que, de acuerdo con H. S. Maghdid y colaboradores [9], una de las principales limitantes del diagnóstico clínico de pacientes COVID mediante estudios radiológicos es la similitud que tiene en el sentido radiológico con la Neumonía Viral.

Del mismo modo, en el caso de la opacidad pulmonar, encontramos resultados mayores al 82 % en las tres métricas, además de un rango de error del 7 %. Aunque no alcanzan los porcentajes de otras categorías, aún se encuentran en un rango aceptable.

Finalmente, en la sección 1 mencionábamos como los modelos *Computer Aided Designs* CAD pos sus siglas en inglés, han demostrado facilitar la labor en el área médica, principalmente en la detección de Cáncer de Mama, nódulos pulmonares, mamogramas, entre otros. Particularmente, en el campo de detección del COVID-19 y Neumonía Viral mediante modelos CAD, se han encontrado resultados de diagnóstico con sensibilidades de hasta 83.61 %, 96.8 % y 80.02 % en el 2020 [12], [17], [18] y [9], resultados que son comparables con el desempeño de un especialista e incluso con los resultados obtenidos por la prueba RT-PCR. Así, encontramos que los resultados obtenidos para el modelo propuesto resultan equiparables con otros en su categoría.

Por lo que podemos concluir que se cumplieron los objetivos de la tesis tras lograr la construcción de un modelo que, mediante el entrenamiento con redes neuronales pre entrenadas, permite la detección de las 4 clasificaciones. Además se determinó que el modelo trabaja mejor tras ser entrenado con la red ResNet50 y los resultados encontrados, aunque no nos hablan por completo del desempeño del modelo, alientan a continuar la línea de investigación.

REFERENCIAS

[1] D. Matich, " Redes Neuronales: Conceptos básicos y aplicaciones," Universidad Tecnológica Nacional, México, vol. 41, pp. 12–16, 2001.

[2] J. Jeong, "The Most Intuitive and Easiest Guide for Convolutional Neural Network," 12 2021.

[3] F. Chollet, Deep Learning with Python. Manning, 2da edición ed., 2021.

[4] T. Torao, "Keras: CNN (/Fine Tuning) - MOXBOX Keras TensorFlow VGG16." https://hazm.at/mox/machine-learning/computer-vision/classification/keras-transfer-learning/index.html, 07 2018.

[5] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He, "Aggregated Residual Transformations for Deep Neural Networks," UC San Diago, Facebook

AI Research. EEUUA., 2017.

[6] H. Wu, M. Xin, W. Fang, H.-M. Hu, and Z. Hu, "Multi-Level Feature Network With Multi-Loss for Person Re-Identification," IEEE Access, vol. 7, no. 1, pp. 91052–91062, 2019.

[7] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks." https://papers.nips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html, 2012.

[8] Kaggle, "COVID-19 chest xray." https://www.kaggle.com/datasets/bachrr/covid-chest-xray, 05 2020.

[9] H. Maghdid, A. Assad, K. Ghafoor, A. Sadiq, and M. Khan, " Diagnosing COVID- 19 Pneumonia from X-Ray and CT Images using Deep Learning and Transfer Learning Algorithms.," Cornell University. EEUUA/Iraq., 2020.

[10] D. Varshni, K. Thakral, L. Agarwal, R. Nijhawan, and A. Mittal, "Pneumonia Detection Using CNN based Feature Extraction," 2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT), 2019.

[11] S. S. Chaturvedi, J. V. Tembhurne, and T. Diwan, "A multi-class skin Cancer classification using deep convolutional neural networks," Multimedia Tools and Applications, vol. 79, no. 39-40, pp. 28477–28498, 2020.

[12] J. Zhang, Y. Xie, G. Pang, Z. Liao, J. Verjans, W. Li, Z. Sun, J. He, Y. Li, C. Shen, and Y. Xia, "Viral Pneumonia Screening on Chest X-rays Using Confidence-Aware Anomaly Detection.," IEEE, Transactions on medical imaging., 2020.

[13] E. Pereira Suárez and et.al., COVID-19: Una visión integral. Aspectos Biológicos y Psicosociales desde el ámbito universitario. Universidad Autónoma de Guadalajara, primera edición ed., 2021.

[14] J. Watson, P. Whiting, and J. Brush, "Interpreting a COVID-19 test result.," British Medical Journal. U.K., vol. 369, pp. 1–7, 2020.

[15] M. Baldomá España, A. Zidan, J. Segura García, N. Planas Gananu, and M. Muñoz Pérez, "Frecuencia de Rx de tórax indicativas de afectación por SARS-CoV-2 de marzo a mayo de 2020 en la población de un área de salud urbana.," Elsevier. España., 2020.

[16] H. Y. F. Wong, H. Y. S. Lam, A. H.-T. Fong, S. T. Leung, T. W.-Y. Chin, C. S. Y. Lo, M. M.-S. Lui, J. C. Y. Lee, K. W.-H. Chiu, T. W.-H. Chung, E. Y. P. Lee, E. Y. F. Wan, I. F. N. Hung, T. P. W. Lam, M. D. Kuo, and M.-Y. Ng, "Frequency and Distribution of Chest Radiographic Findings in Patients Positive for COVID-19," Radiology, vol. 296, no. 2, pp. E72–E78, 2020.

[17] S. L. Mayanga Sausa, R. Guerra Tueros, D. Lira Villasante, and D. Pastor Gutiérrez, "Utilidad de la radiografía de tórax en el cntexto de la pandemia por Sars- Cov-2.." http://www.scielo.org.pe/

scielo.php?script=sci_arttext&pid=S2308-05312020000400682, 2020.

[18] P. P. Teixeira e Silva Torres, K. Loureiro Irion, and E. Marchiori, "COVID-19: chest X-rays to predict clinical outcomes," Jornal Brasileiro de Pneumologia, vol. 46, no. 5, pp. e20200464–e20200464, 2020.

[19] A. Tahamtan and A. Ardebili, "Real-time RT-PCR in COVID-19 detection: issues affecting the results," Expert Review of Molecular Diagnostics, vol. 20, no. 5, pp. 453–454, 2020.

[20] W. McCulloch and W. Pitts, "A logical Calculus of the ideas immanent in nervous activity," Bulletin of Mathematical Biology, pp. 99–115, 1990.

[21] R. Hecht-Nielsen, "Perceptrons. Technical Report No. 0403," University California San Diego. Institute for Neural Computation., 2004.

[22] M. Nielsen, Neural Networks and Deep Learning . Determination Press, 2015.

[23] D. Anderson and G. McNeill, " Artificial neural networks technology," Kaman Sciences Coorporation, vol. 258, no. 6, pp. 1–83, 1992.

[24] R. Calvo and S. D'Mello, "Affect detection: an interdisciplinary review of models, methods and their apllications.," IEEE Trans Affect Comput, vol. 1, no. 1, pp. 18–37, 2010.

[25] J. Howard and S. Gugger, Deep Learning for Coders with fastai & PyTorch. AI applications Without a PhD. O'Reilly, 2020.

[26] C. Albon, Machine Learning with Python Cookbook: Practical Solutions from Preprocessing to Deep Learning. O'Reilly, 2018.

[27] D. Cirergan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classifcation," IEEE conference on computer vision and pattern recognition, pp. 3642–3649, 2012.

[28] H. Lefteri and et. al., Fuzzy and neural approaches in engineering. Wiley SONS INC., 1997.

[29] J. Huang and C. Ling, "Using AUC and accuracy in evaluating learning algorithms," IEEE Transactions on Knowledge Data Engineering, vol. 17, pp. 299–310, 2005.

[30] E. Petersen and et. al., "Image processing with neural networks—a review," Pattern recognition, vol. 35, no. 10, pp. 2279–2301, 2002.

[31] O. Russakovsky, "ImageNet Large Scale Visual Recognition Challenge." https://arxiv.org/abs/1409.0575, 09 2014.

[32] M. Talo, O. Yildirim, U. B. Baloglu, G. Aydin, and U. R. Acharya, "Convolutional neural networks for multi-class brain disease detection using MRI images," Computerized Medical Imaging and Graphics, vol. 78, p. 101673, 2019.

[33] C. Szegedy, W. Liu, J. Yangqing, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhocke, and A. Ravinovich, " Going deeper with convolutions," Google Inc., 2014.

[34] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classifcation with deep convolutional neural networks.," Advances in neural information processing systems, pp. 1097–1105, 2012.

[35] S. Marsland, Machine Learning: An Algorithmic Perspective. Chapman, 2014.

[36] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," Google Inc. California, E.E.U.U.A., 2015.

[37] Z. Hoo, J. Candlish, and D. Teare, "What is an ROC curve." https://emj.bmj.com/content/34/6/357.citation-tools, 2017.

[38] Artificial Intelligence-All in one, "Lecture 6.7 — Logistic Regression — MultiClass Classification OneVsAll — [Andrew Ng]." https://www.youtube.com/watch?v=-EIfb6vFJzc, 01 2017.

[39] W. J. Youden, "Index for rating diagnostic tests." https://doi.org/10.1002/1097-0142(1950)3:1<32::AID-CNCR2820030106>3.0.CO;2-3, 1950.

[40] A. Jakimowicz, "The Role of Entropy in the Development of Economics," Entropy, vol. 22, no. 4, p. 452, 2020.

[41] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," Computer Vision and Pattern Recognition, 2015.

[42] M. Gálvez M and C. Montoya M, "Error en el informe radiológico: La paradoja del elefante en la habitación y otros tropiezos," Revista chilena de radiología, vol. 23, no. 2, pp. 80–89, 2017.

[43] "ImageNet." https://www.image-net.org/challenges/LSVRC/2016/index.php, 2020.