Proyecto 2
Redes Neuronales

1st Camacho Hernández José Julián
Instituto Tecnológico de Costa Rica
Reconocimiento de Patrones
Cartago, Costa Rica
jcamacho341@estudiantec.cr

2nd Guillén Fernández José Leonardo
Instituto Tecnológico de Costa Rica
Reconocimiento de Patrones
Cartago, Costa Rica
leoguillen@estudiantec.cr

***Resumen*—This paper will explain and show the performance of two different neural networks using the COVID-19 Radiography Dataset. The main objective is to predict whether the patient in each image of the test set is healthy or is diagnosed with COVID-19, Lung opacity or Viral Pneumonia. At first, Multilayer perceptron will be trained and tested using the raw dataset and then applying a feature extractor filter to it. The same analysis will be done using a Convolutional Network and applying bilateral filter to the image. The results of using raw and filtered data will be compared by computing some metrics like accuracy, recall or F1 score. Results show that networks perform better when filters or feature extractors are applied to the dataset.**

***Index Terms*—Machine Learning, Neural Network, Multilayer Perceptron (MLP), Convolutional Networks (CNN), X-Ray, feature extractor, heat maps**

I. Introducción

El siguiente informe se basa en el análisis del comportamiento de dos tipos de redes neuronales: los perceptrones multicapa y las redes convolucionales. Para realizarlo, se utilizará el set de datos COVID-19 Radiography Dataset, y se realizarán entrenamientos y pruebas sobre los modelos utilizando el conjunto de datos normal y también aplicando ciertos filtros como el bilateral o el feature extractor de *Local Binary Patterns*.

El objetivo principal del *paper* es implementar estos tipos de redes neuronales con el fin de solucionar un problema de clasificación y comparar así los resultados de utilizarlas con datos filtrados y sin filtrar. Se analizarán también las áreas de mayor influencia para las decisiones y predicciones de los modelos.

Se inicia explicando la teoría de cada modelo.

I-A. Perceptrón Multicapa

Los perceptrones multicapa (también conocidos como MLP, por sus siglas en inglés: Multi-Layer Perceptron) son una arquitectura fundamental en los campos relacionados con del aprendizaje supervisado. Son modelos de redes neuronales artificiales compuestas por múltiples capas con variable cantidad de neuronas o nodos interconectados, que han demostrado ser altamente eficaces en una amplia gama de tareas de aprendizaje automático.

En un perceptrón multicapa, cada neurona procesa las entradas, y luego las pasa a través de una función de activación preferiblemente no lineal, entre las cuales se destacan *sigmoid, ReLU, tanh, softmax*, entre otras. La presencia de capas ocultas entre la capa de entrada y la capa de salida permite que estas redes sean capaces de modelar relaciones no lineales complejas en los datos de entrada. [1]

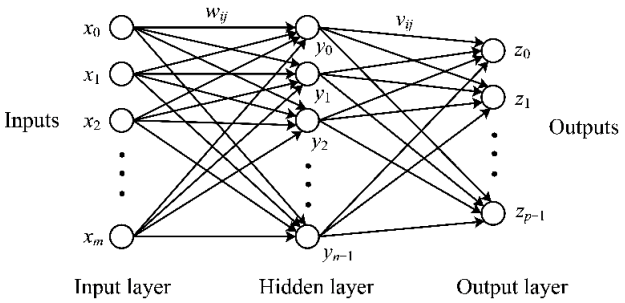

Figura 1. Arquitectura de un perceptrón multicapa.

Durante el entrenamiento de un MLP, se utiliza un algoritmo de optimización, como el descenso del gradiente, para ajustar los pesos de las conexiones entre las neuronas. El objetivo es minimizar la función de costo que mide la diferencia entre las salidas obtenidas o predichas, y las salidas esperadas.

Este método de aprendizaje se basa en la primera propagación de los datos de entrada a través de la red hacia adelante y la propagación hacia atrás del error para calcular los gradientes y corregir los pesos.

La capacidad de los perceptrones multicapa para aprender patrones complejos y patrones de información, así como su flexibilidad para adaptarse a una variedad de tareas, los ha convertido en una herramienta poderosa en el campo del aprendizaje supervisado y automático [2].

La importancia y eficacia de los multiperceptrones se demuestra en su amplia aplicación en diversas áreas, como la calidad de imagen [3], el reconocimiento de voz [4] et al. 2012), el procesamiento del lenguaje natural [5], entre otros.

I-B. Redes Convolucionales

Las redes neuronales convolucionales (CNN) son un tipo de arquitectura de aprendizaje profundo ampliamente utilizada en el procesamiento de imágenes. Estas redes están diseñadas específicamente para reconocer patrones y características en

datos visuales, lo que las hace ideales para analizar imágenes radiográficas en el contexto de la detección de COVID-19.

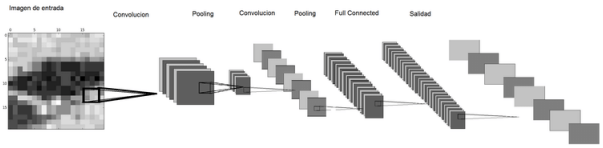

Figura 2. Arquitectura de una Red Convolucional.

El conjunto de datos COVID-19 Radiography Dataset [6, 7, 8, 9] proporciona una colección de imágenes radiográficas de tórax, tanto de pacientes con COVID-19 como de individuos sanos. Estas imágenes se utilizan para entrenar y evaluar la CNN [6, 7, 9] en la tarea de clasificación binaria: identificar si una imagen es positiva para COVID-19 o negativa para COVID-19.

En cuanto al procesamiento de las imágenes radiográficas, una técnica comúnmente utilizada es el filtro bilateral. El filtro bilateral es un operador de suavizado que conserva los bordes en una imagen al tiempo que reduce el ruido y las imperfecciones. A diferencia de otros filtros de suavizado, el filtro bilateral tiene en cuenta tanto la proximidad espacial como las diferencias de intensidad entre píxeles, lo que permite preservar detalles importantes mientras se elimina el ruido.

Cuando se aplica el filtro bilateral a las imágenes radiográficas del conjunto de datos COVID-19 Radiography Dataset, se espera que resalte las características clave asociadas con el COVID-19, como las opacidades pulmonares o las infiltraciones. Esto puede mejorar la capacidad de la CNN para detectar y clasificar correctamente las imágenes.

En contraste, si las imágenes radiográficas se procesan sin el filtro bilateral, se pueden perder algunos detalles sutiles debido al ruido y las imperfecciones presentes en las imágenes originales. Esto podría afectar la precisión y eficacia de la CNN en la detección y clasificación del COVID-19.

Por lo tanto, la comparación entre el uso del filtro bilateral y el procesamiento sin filtro en la CNN es crucial para evaluar la influencia de esta técnica de suavizado en la capacidad de la red para identificar características relevantes y tomar decisiones precisas de clasificación. Los resultados de esta comparación pueden proporcionar información valiosa para mejorar los sistemas de detección y diagnóstico del COVID-19 basados en imágenes radiográficas.

II. Detalles del diseño del programa desarrollo

En la presente sección se explica el diseño de las redes neuronales implementadas.

II-A. Perceptrón Multicapa

A continuación se visualizan y explican los pasos en el funcionamiento de la red MLP desarrollada.

- **Feature engineering** Se añade este paso para manejar oportunamente los datos que la red utilizará. Puede incluir extracción de *features*, normalización de los datos, entre otros.
- **Definir arquitectura de la MLP** Esto implica determinar el número de capas ocultas y el número de neuronas en cada capa oculta.
- **Inicializar los pesos de manera aleatoria** Se inicializan aleatoriamente para comenzar el proceso de entrenamiento.
- **Entrenar la red neuronal** En este punto se marca el inicio del entrenamiento, El objetivo es minimizar el error entre las salidas predichas por la red y las salidas esperadas.
- **Repetir hasta que se cumpla el criterio de parada** El entrenamiento se realiza en iteraciones repetidas hasta que se cumpla un criterio de parada, que en este caso será cierta cantidad de *epochs*.
- **Propagación hacia adelante (Forward Propagation)** Cada muestra se pasa a través de la red neuronal para calcular las salidas de cada neurona en cada capa oculta y en la capa de salida. Esto implica aplicar la influencia de los pesos, y luego aplicar una función de activación a cada neurona.
- **Propagación hacia atrás (Backward Propagation)** Después de obtener las salidas de la red, se calcula el error en la capa de salida al comparar las salidas predichas con las salidas esperadas. Luego, el error se propaga hacia atrás a través de la red para calcular los errores en las capas ocultas.
- **Actualizar los pesos** Utilizando el algoritmo de retropropagación del error, se actualizan los pesos y sesgos de la red para reducir el error en las salidas. Esto implica ajustar los valores de los pesos.
- **Predecir** Una vez finalizado el entrenamiento, la red neuronal se puede utilizar para hacer predicciones en nuevos datos.

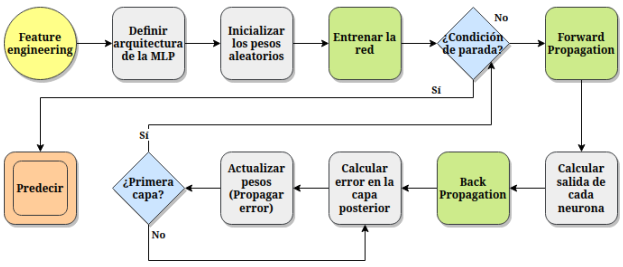

Figura 3. Diagrama de Flujo de un Perceptrón Multicapa.

Seguidamente se muestran los diseños de arquitecturas propuestos para este tipo de redes neuronales.

II-A1. Diseño de arquitectura 1: El primer diseño de arquitectura es el que se muestra en la figura 4.

Consta de dos capas ocultas con función de activación ReLU debido a que con esa se obtuvieron los mejores resultados. La capa de entrada para todas estas arquitecturas es de 67600. Este corresponde al número de píxeles en una imagen $260 \times$

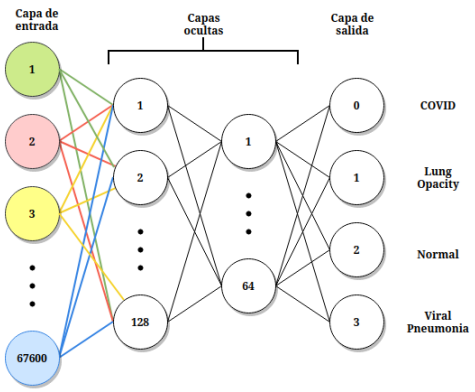

Figura 4. Diseño 1 de arquitectura de MLP.

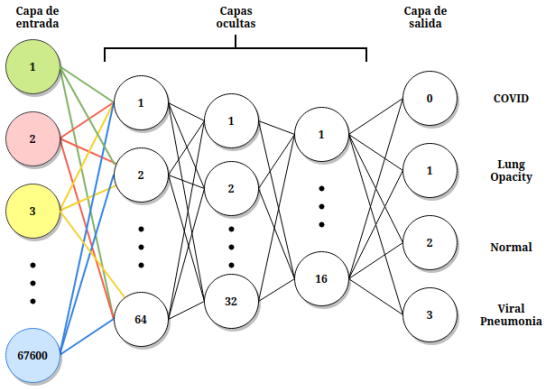

Figura 6. Diseño 3 de arquitectura de MLP.

260, ya que a las imágenes del set de datos de *COVID-19 Radiography Dataset* se les realiza un recorte de los bordes, porque se considera que no aportan información relevante al modelo.

Para este caso, se elige una primera capa oculta de 128 neuronas, seguida de una de 64. Finalmente, se tiene la capa de salida, que utiliza la función *softmax* para predecir según las probabilidades. Esta última capa es de 4 neuronas debido a que el problema busca clasificar las imágenes en esta cantidad de grupos.

II-A2. Diseño de arquitectura 2: El diseño 2 consta de una única capa oculta, como se presenta en la figura 5, esta vez de 64 neuronas. Se realizaron pruebas con mayor cantidad de neuronas y no se obtuvieron los mejores, por lo que se elige esta configuración para evaluar el caso de una sola capa oculta.

La capa oculta utiliza la función ReLU para su activación, y las capas de entrada y salida se mantienen como se detalló anteriormente.

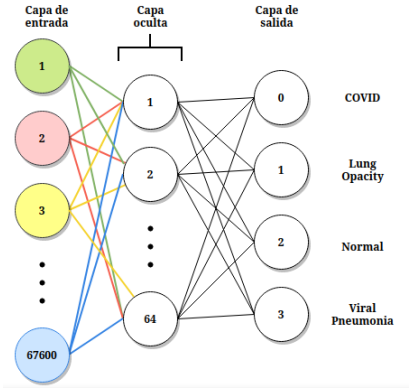

Figura 5. Diseño 2 de arquitectura de MLP.

II-A3. Diseño de arquitectura 3: Como diseño número 3 se presenta el siguiente:

En este caso se cuenta con tres capas ocultas, de 64, 32 y 16 neuronas respectivamente. Se elige esta arquitectura para evaluar el caso en que se van reduciendo el número de neuronas y se acerca a la cantidad que se tienen en la capa de salida.

II-B. Redes Convolucionales

En la siguiente figura se presenta el diagrama de flujo del diseño de una red convolucional.

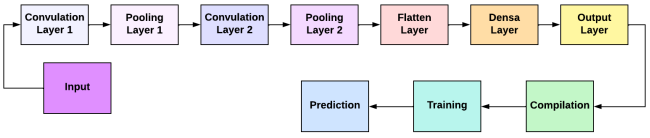

Figura 7. Diagrama de Flujo de una Red Convolucional.

- **Entrada**: Esta etapa representa las imágenes radiográficas que se utilizarán como datos de entrada para el modelo de CNN.
- **Capa Conv2D**: Es una capa convolucional que realiza operaciones de convolución en las imágenes de entrada. Cada filtro en esta capa extrae características específicas de la imagen, y el número de filtros se determina mediante el parámetro ”Número de filtros (X)”. El tamaño del kernel determina el tamaño de la ventana de convolución que se desliza sobre la imagen. La función de activación ReLU se aplica a los mapas de características resultantes para introducir no linealidad. El relleno ”same.ªsegura que el tamaño de salida sea el mismo que el de la imagen de entrada.
- **Capa Pooling**: Es una capa de submuestreo que reduce la dimensión espacial de los mapas de características obtenidos de la capa convolucional. El tamaño de pool (Z) define el tamaño de la ventana de agrupación que se desliza sobre la imagen y selecciona el valor máximo o promedio de cada región, dependiendo del tipo de pooling utilizado.
  Repetir capas convolucionales y de pooling: Estas capas se repiten según sea necesario para construir la arquitectura de la CNN. Añadir más capas convolucionales y de pooling puede ayudar a extraer características más complejas y abstractas de las imágenes.

- **Capa Flatten**: Esta capa se utiliza para aplanar la salida de la última capa convolucional, lo que significa convertir la salida en un vector unidimensional. Prepara los datos para ser alimentados en una capa densa.
- **Capa Densa**: Es una capa completamente conectada en la que cada neurona está conectada a todas las neuronas de la capa anterior. El número de neuronas se determina mediante el parámetro ”Número de neuronas (N)”. La función de activación ReLU se aplica a las salidas de las neuronas.
- **Capa de Salida**: Es la capa final del modelo, que produce la salida deseada. En este caso, hay una sola neurona en la capa de salida y la función de activación utilizada es la sigmoide. Esto se debe a que se trata de una clasificación binaria (COVID-19 o no COVID-19).
- **Compilación**: Esta etapa configura el modelo para el entrenamiento, especificando la función de pérdida y el optimizador. La función de pérdida mide qué tan bien el modelo se ajusta a los datos de entrenamiento, y el optimizador ajusta los pesos del modelo para minimizar la pérdida.
- **Training**: Aquí es donde el modelo se entrena utilizando los datos de entrenamiento. Durante el entrenamiento, el modelo ajusta sus parámetros iterativamente para mejorar su rendimiento en la tarea de clasificación.
- **Predicción**: Después del entrenamiento, el modelo se evalúa utilizando los datos de prueba para medir su rendimiento en datos no vistos previamente. Esto proporciona una evaluación objetiva de la capacidad del modelo para generalizar y clasificar correctamente nuevas imágenes radiográficas.

II-B1. Diseño de arquitectura 1: **Arquitectura propia:** ***Capas Convolucionales:***

La primera capa *Conv2D* tiene 32 filtros de tamaño 3x3 y utiliza la función de activación ReLU. Esta capa realiza la convolución de los datos de entrada en busca de características relevantes en la imagen.

La capa *MaxPooling2D* se utiliza para reducir la dimensionalidad espacial de las características extraídas por la capa convolucional anterior. En este caso, se utiliza un tamaño de ventana de 2x2 para realizar el muestreo máximo. ***Capas Totalmente Conectadas:***

La capa *Flatten* se utiliza para aplanar las características obtenidas de la capa anterior, convirtiéndolas en un vector unidimensional. Esto permite la conexión total entre las capas convolucionales y las capas densas posteriores.

La capa *Dense* con 64 neuronas utiliza la función de activación ReLU. Esta capa tiene conexiones completamente conectadas con la capa anterior y aplica una transformación no lineal a los datos.

Características Clave: La arquitectura utiliza convoluciones para extraer características locales de las imágenes, capturando patrones como bordes, texturas y formas.

El muestreo máximo reduce la dimensionalidad espacial de las características, ayudando a disminuir la cantidad de parámetros y a obtener una representación más compacta.

Las capas densas finales realizan la clasificación de las características extraídas en las clases de salida utilizando la función de activación softmax, lo que permite obtener una distribución de probabilidades para cada clase.

II-B2. Diseño de arquitectura 2: **Arquitectura InceptionV3:**

Módulos de Inception: La arquitectura InceptionV3 utiliza módulos de Inception, que son bloques de operaciones convolucionales paralelas con diferentes tamaños de filtro. Estos módulos permiten capturar características a diferentes escalas espaciales y combinarlas en una representación conjunta.

Capas convolucionales: InceptionV3 tiene múltiples capas convolucionales, incluyendo capas convolucionales 1x1, 3x3, 5x5 y capas de agrupamiento máximo. También utiliza capas de ”bottleneck”para reducir la dimensionalidad antes de aplicar capas convolucionales más grandes.

Características clave: La arquitectura InceptionV3 es conocida por su capacidad para capturar características a diferentes escalas y su eficiencia computacional. Ha sido ampliamente utilizada para la clasificación de imágenes y la extracción de características en tareas de visión por computadora.

III. Resultados y análisis

En la presente sección, se detallan aspectos relacionados con los resultados obtenidos al llevar a cabo el ciclo de diseño de un problema de reconocimiento de patrones.

III-A. Perceptrón Multicapa sin feature extractor

Para el MLP, en primer lugar, se realizará el entrenamiento y predicciones con el conjunto de datos sin algún tipo de filtro previo.

Como parte del *feature engineering*, se normalizan los datos y se recortan píxeles en los bordes de las imágenes. Esto debido a que como los puntos de interés del modelo se encuentran en los pulmones, por lo que parte de los hombros o cuello de las personas se considera como ruido para el modelo.

A continuación se presentan los resultados, tanto para el conjunto de prueba como de entrenamiento, para la arquitectura propuesta que mejores resultados produjo.

III-A1. Diseño de arquitectura 1: En esta primera arquitectura, para el conjunto de prueba se obtienen las métricas que se presentan en la figura 8.

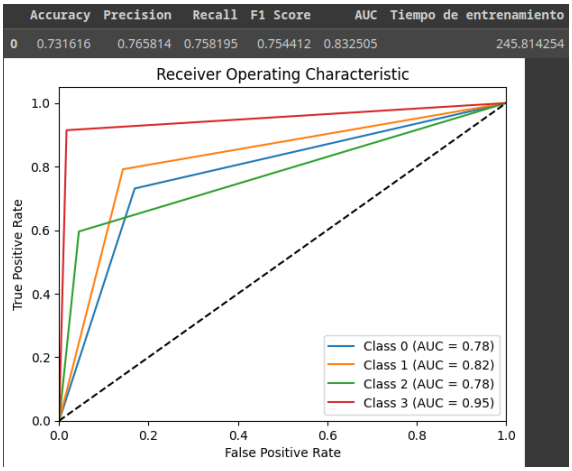

Figura 8. Resultados para el conjunto de prueba.

Se presta singular importancia al *recall*, ya que como se trata de un problema de predicción médica, se debe analizar la tasa de falsos negativos. En este caso se obtiene 0.75 en esta métrica y valores cercanos en las demás que se derivan de la matriz de confusión.

Se obtienen 4 gráficas de las ROC, una para cada clase, esto debido a que se toma un enfoque de todos contra uno. Esto quiere decir que para cada clase se modela como un problema de clasificación binaria, donde dicha clase es el positivo y las demás el negativo.

Para este caso se obtienen valores de AUC mayores a en todas las clases indicando la efectividad del modelo.

Los siguientes resultados son para el conjunto de entrenamiento.

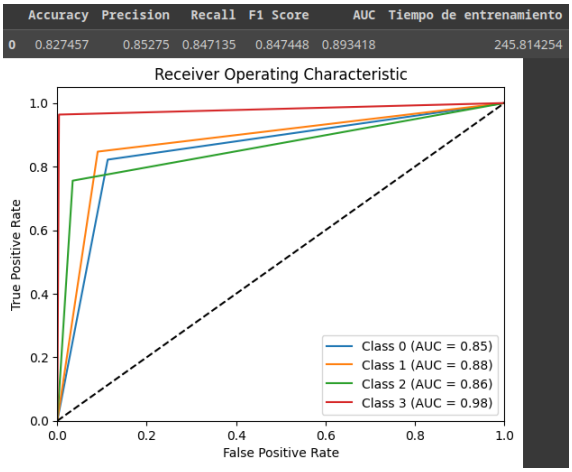

Figura 9. Resultados para el conjunto de entrenamiento.

De estos se analiza, que se obtienen valores de métricas mayores a los del conjunto de pruebas, por lo que se presenta un ligero *overfitting*.

Esta arquitectura tuvo un tiempo de entrenamiento de aproximadamente 4 minutos y 6 segundos.

III-A2. Diseño de arquitectura 2: Seguidamente se presentan las métricas para la segunda arquitectura propuesta.

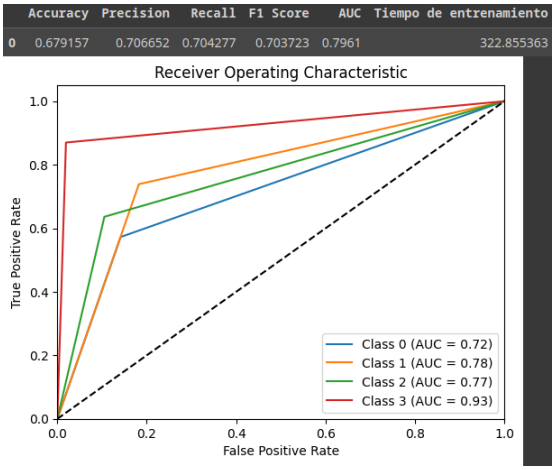

Figura 10. Resultados para el conjunto de prueba.

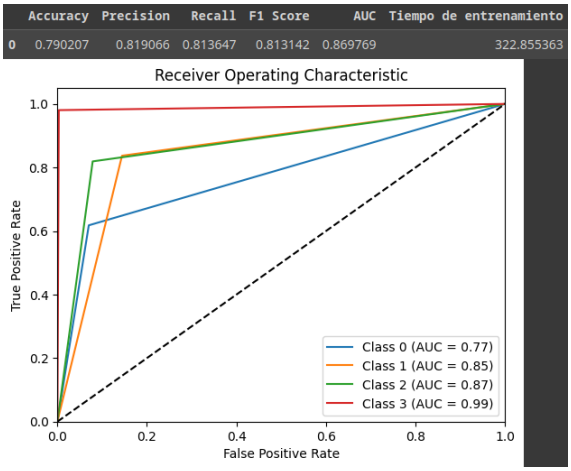

Figura 11. Resultados para el conjunto de entrenamiento.

Se observa que las métricas obtenidas son ligeramente menores a la de la arquitectura anterior, pero siguen la tendencia de un ligero *overfitting*.

Esta arquitectura tuvo un tiempo de entrenamiento de aproximadamente 5 minutos y 23 segundos.

III-A3. Diseño de arquitectura 3: A continuación, en las figuras 12 y 13 se presentan las métricas para la arquitectura propuesta número 3.

Esta arquitectura tuvo un tiempo de entrenamiento de aproximadamente 3 minutos y 20 segundos.

III-B. Perceptrón Multicapa con feature extractor

Seguidamente, se aplica un *feature extractor* a las muestras en el conjunto de datos. Para este caso se utiliza el algoritmo *Local Binary Patterns (LBP)*, que es un método que se utiliza para describir las características de textura de las imágenes. Esto con el fin de acentuar las características que el algoritmo considera más importantes en la imagen para que el modelo se enfoque mejor en ellas.

Para aplicar el filtro se utilizó la biblioteca *skimage*. La función que realiza el *feature extractor* recibe los parámetros

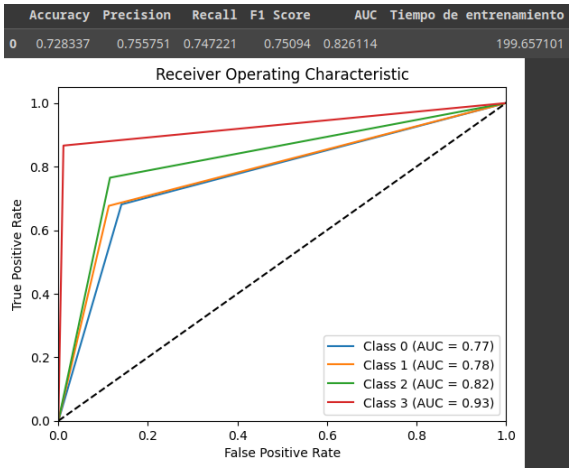

Figura 12. Resultados para el conjunto de prueba.

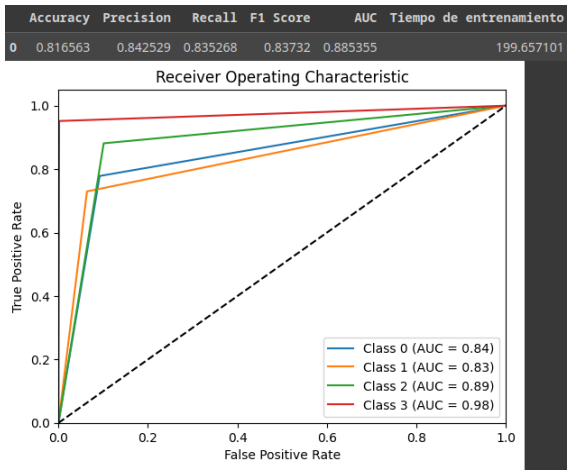

Figura 13. Resultados para el conjunto de entrenamiento.

de radio y número de puntos por analizar. Se realizaron múltiples pruebas con diferentes combinaciones de estos parámetros y los mejores resultados se obtuvieron con un radio de 3 y número de puntos de 21. El filtrado con esos parámetros se visualiza de la siguiente manera:

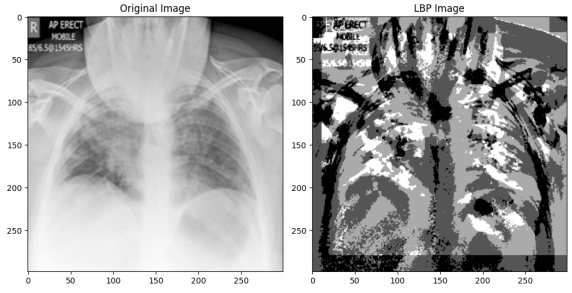

Figura 14. Filtrado de las imágenes por medio de LBP.

A continuación se presentan los resultados, tanto para el conjunto de prueba como de entrenamiento, para arquitectura propuesta que mejores resultados arrojó.

III-B1. Diseño de arquitectura 3: La arquitectura que mejores resultados arrojó fue la número 3, compuesta por 3 capas ocultas. Durante las múltiples pruebas realizadas para el set de datos con *feature extractor*, se observó que las mejores métricas se obtienen al definir la arquitectura con un mayor número de capas ocultas con baja cantidad de neuronas, en lugar de aumentar la cantidad de neuronas y reducir el número de capas.

Las mejores métricas obtenidas para el conjunto de pruebas son las que se presentan en la siguiente figura 15.

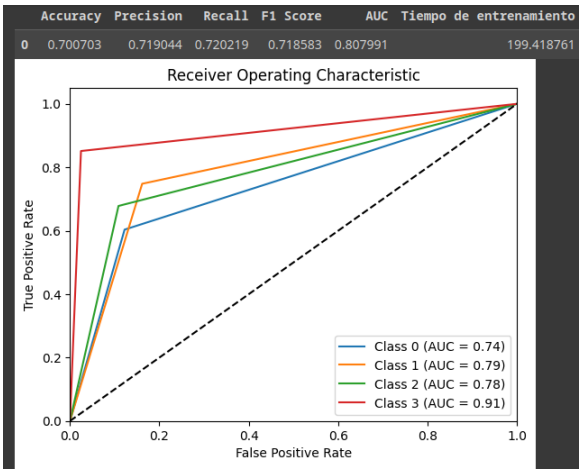

Figura 15. Resultados para el conjunto de prueba.

En la figura 16 se presentan las métricas obtenidas para el conjunto de entrenamiento.

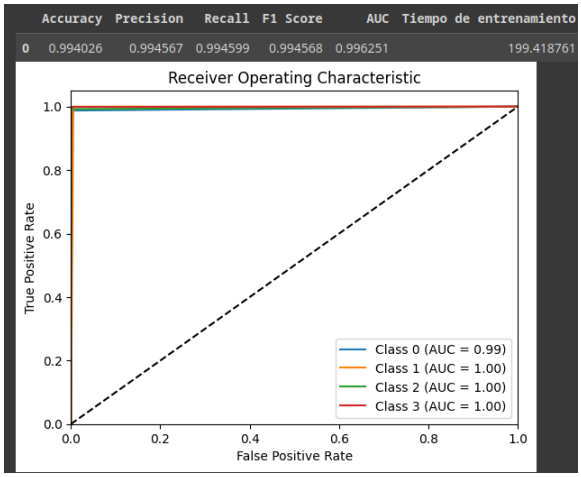

Figura 16. Resultados para el conjunto de entrenamiento.

De los resultados se analiza que el modelo presenta un caso marcado de *overfitting*. Esto ya que el modelo es exageradamente bueno entrenando, pero en las pruebas no logra obtener métricas tan buenas, lo que indica que la red se ajustó tanto a los datos del *training* que no logra clasificar de manera excelente.

Adicionalmente, es importante apuntar que los resultados presentan un caso contrario a lo que se esperaba, ya que

teóricamente los *features extractors* se utilizan para destacar los aspectos más relevantes de las imágenes. Esto con el objetivo de que los modelos se centren en dichas características para así predecirde manera más óptima.

Sin embargo, los resultados obtenidos muestran un empeoramiento de las métricas de predicción, por lo que no fue posible visualizar el efecto real de la aplicación del *feature extractor* en el set de datos.

Respecto a esto, se deben considerar otros acercamientos para aplicar el filtro utilizado u otros, para así intentar obtener mejores predicciones.

Esta arquitectura tuvo un tiempo de entrenamiento de aproximadamente 3 minutos y 20 segundos.

IV. Perceptrón Multicapa: Métricas

- **¿Cuál es el tamaño de los modelos?**
  Los modelos de MLP tiene los siguientes tamaños:
  - Diseño 1: $67600 \times 128 \times 64 \times 4$ Al realizar los cálculos por medio de la siguiente fórmula:

    $$(67600 * 128) + 128 + (128 * 64) + 64 + (64 * 4) + 4$$

    para este modelo se calculan: 8 661 444 de parámetros entre pesos y sesgos.
  - Diseño 2: $67600 \times 64 \times 4$ Al realizar los cálculos, para este modelo se calculan: 4 326 724 de parámetros entre pesos y sesgos.
  - Diseño 3: $67600 \times 64 \times 32 \times 16 \times 4$ Al realizar los cálculos, para este modelo se calculan: 4 329 140 de parámetros entre pesos y sesgos.
- **¿Cuál es la arquitectura (cuáles y cuántas capas se utilizaron)?**
  Todos los modelos MLP utilizaron función de activación ReLU porque dio los mejores resultados. Estas son las arquitecturas establecidas en cada caso:
  - Diseño 1: Una capa de entrada de 67600 neuronas, dos capas ocultas de 128 y 64 respectivamente. Capa de salida de 4 neuronas.
  - Diseño 2: Una capa de entrada de 67600 neuronas, una capa oculta de 64 y una capa de salida de 4 neuronas.
  - Diseño 3: Una capa de entrada de 67600 neuronas, tres capas ocultas de 64, 32 y 16 respectivamente. Capa de salida de 4 neuronas.
- **¿Qué ventajas tiene utilizar un feature extractor? referirse a las métricas y tiempos**
  Particularmente, en este caso no se observó una diferencia marcada al utilizar el *feature extractor*. para el conjunto de pruebas. De hecho para este las métricas resultaron incluso un poco peores, por ejemplo el *recall* bajó 0.03.
  Sin embargo, sí se nota una gran diferencia al realizar las predicciones con el conjunto de entrenamiento, ya que para el modelo con *feature extractor* se obtuvieron métricas casi perfectas llegando a valores muy cercanos a 1.
  Respecto a los tiempos, si bien no se notó una diferencia marcada en el entrenamiento, el tiempo de filtrado de las imágenes sí fue significativo.
- **¿Qué modelo se entrena más rápido?**
  En términos generales los modelos tanto con *feature extractor* como sin él, se entrenaron en tiempos que pueden ser considerados similares, ya que difieren en unos cuantos segundos.
  Respecto a las distintas arquitecturas, la que entrenó más rápido fue la 3.
- **¿Cuál fue el mejor modelo?**
  A partir de las métricas, se puede concluir que el modelo con la arquitectura número 3, compuesto por tres capas ocultas de 64, 32 y 16 neuronas, tanto aplicando el *feature extractor* y sin él, fue el mejor.

IV-A. Redes Convolucionales sin filtro

En el modelo sin filtro se puede observar que con la matriz de confusión es posible generar varias métricas. Sin embargo, también se generan en sus respectivas tablas, donde se aprecia que al procesar las imágenes sin filtro se tiene una leve mejora en las métricas con la predicción con el conjunto de entrenamiento.

El análisis nos indica la presencias de *overfitting*, mostrándonos que todas las demás métricas son superiores, por lo que se deben aplicar técnicas de regulación para evitar el sobreajuste a los datos de entrenamiento.

Accuracy	Precision	Recall	F1-score
0.79	0.88	0.88	0.86

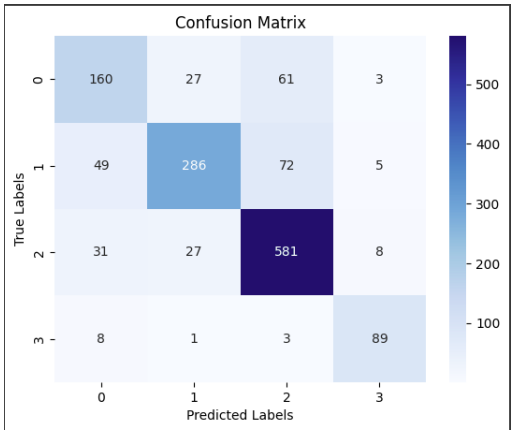

Figura 17. Matriz de Confusión con Test Images

Accuracy	Precision	Recall	F1-score
0.86	0.91	0.95	0.93

IV-B. Redes Convolucionales con filtro

n la figura 19, se muestra como es el filtro bilateral, que lo que hace es suavizar los píxeles haciendo que se vea borroso, el filtro fue aplicado a todo el set de datos para poder predecir

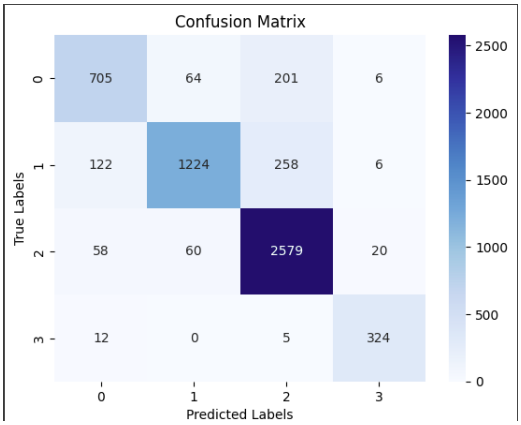

Figura 18. Matriz de Confusión con Train Images

con el modelo establecido y obtener las siguientes métricas, las cuales de la misma manera que el modelo sin filtro, hay *overfitting*.

Sin embargo, en este caso es menor la diferencia entre las métricas de las predicciones del modelo con el set de datos de entrenamiento con el set de datos de prueba, por lo que de la misma manera se debe aplicar técnicas para evitar el sobre ajuste de datos.

En este set de datos específico nos centramos en preferir los falsos positivos por lo que nos estamos centrando en AUC que con la matriz de confusión se sacar el valor de falsos positivo con la fórmula FN/FN+VP donde FN es el número de falsos negativos y VP es el número de verdaderos positivos .

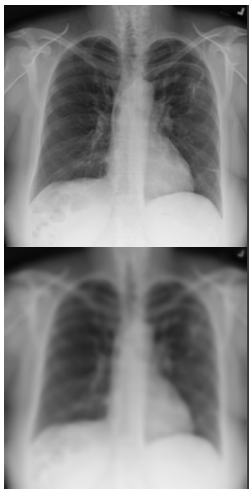

Figura 19. Filtro Bilateral

Accuracy	Precision	Recall	F1-score
0.72	0.76	0.84	0.80

Accuracy	Precision	Recall	F1-score
0.79	0.88	0.88	0.86

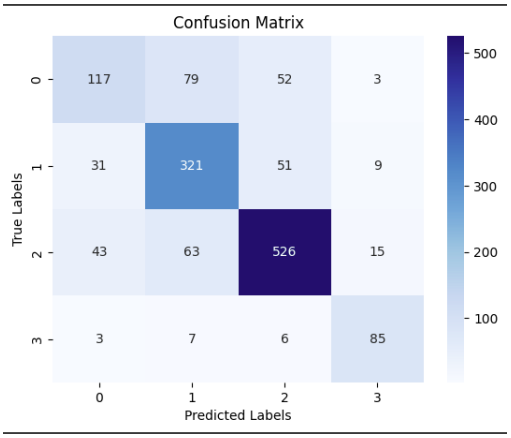

Figura 20. Matriz de Confusión con Test Images con Filtro bilateral

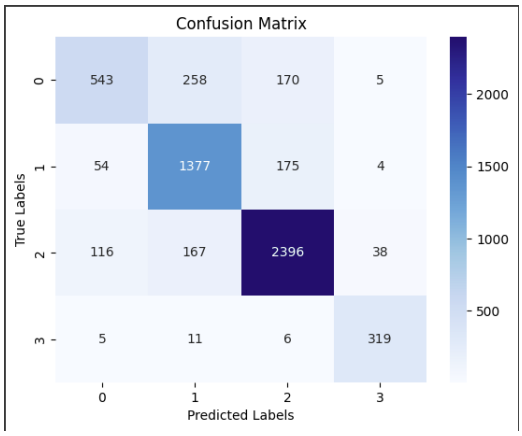

Figura 21. Matriz de Confusión con Train Images con Filtro bilateral

V. Redes Convolucionales: Métricas

- **¿Cuál es el tamaño de los modelos?** Para estas redes se computaron en total 15 297 284 de parámetros.
- **¿Cuál es la arquitectura (cuáles y cuántas capas se utilizaron)?**
  A continuación se detalla la arquitectura utilizada para la CNN:

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 257, 257, 32)	320
max_pooling2d (MaxPooling2D)	(None, 128, 128, 32)	0
conv2d_1 (Conv2D)	(None, 126, 126, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 63, 63, 64)	0
conv2d_2 (Conv2D)	(None, 61, 61, 64)	36928
flatten (Flatten)	(None, 238144)	0
dense (Dense)	(None, 64)	15241280
dense_1 (Dense)	(None, 4)	260

Figura 22. Capas de CNN

- **¿Qué modelo se entrena más rápido?**

El modelo sin filtro bilateral fue el que se entrenó más rápido.

- **¿Cuáles secciones de la imagen son las más importantes para la elección?**
  Depende de la predicción, en general, la parte de los pulmones que es donde se presentan las enfermedades en estudio. Para el caso de un paciente sano se centró en la tráquea, aunque esto puede variar según la muestra.
- **¿Cuál fue el mejor modelo?**
  En términos generales, el modelo sin filtro bilateral se puede considerar como el mejor.

VI. Métricas Generales

- **¿Cuál es el tamaño de los modelos?**
  Los tamaños de los modelos se detallan en las subsecciones IV y V de métricas para cada red neuronal.
- **¿Cuál es la arquitectura (cuáles y cuántas capas se utilizaron)?**
  Se detallan los casos particulares en las subsecciones IV y V.
- **¿Qué modelo se entrena más rápido?**
  Entre todos los modelos evaluados en el presente *paper*, la arquitectura 3 de MLP con *feature extractor* fue la que entrenó más rápido.
- **¿Cuáles secciones de la imagen son las más importantes para la elección?**
  Se detalla en la sección anterior.
- **¿Cuál fue el mejor modelo?**
  En términos generales comparando todos los modelos propuestos en el *paper*, el modelo de CNN sin filtro bilateral se puede considerar como el mejor.

VII. Visualización de mapas de calor

En la presente sección se visualizan las zonas de las imágenes para cada clase que más influencia tuvieron para el modelo en la toma de decisiones respecto a la clasificación final.

Se observa que las zonas de más influencia se encuentran en los pulmones, aspecto que corresponde con la teoría debido a que las enfermedades del set de datos están relacionadas con estos órganos.

Uno de los puntos a recalcar es que, en los mapas de calor, se tienen en los clasificados como Normal, sus pixeles mas importantes en la zona central de torax, mientras el COVID se ve identificado en los pulmones en casi su totalidad, el LUNG OPACITY tiene el punto de referencia en el costado izquierdo del torax y finalmente el Viral Pneumonia varia en la zona baja del torax y en ciertas ocaciones inluyendo un poco los costados.

VII-A. COVID

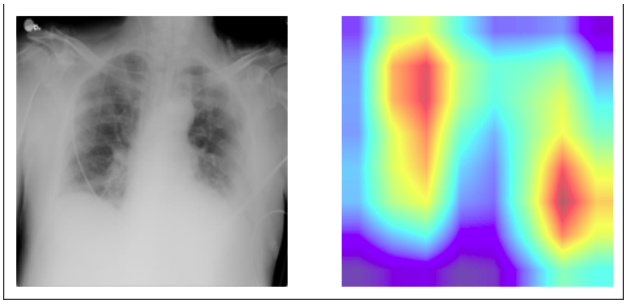

Figura 23. Mapa de Calor de COVID

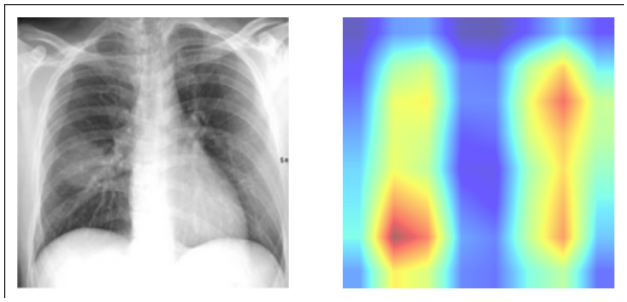

Figura 24. Mapa de Calor de COVID

VII-B. Lung Opacity

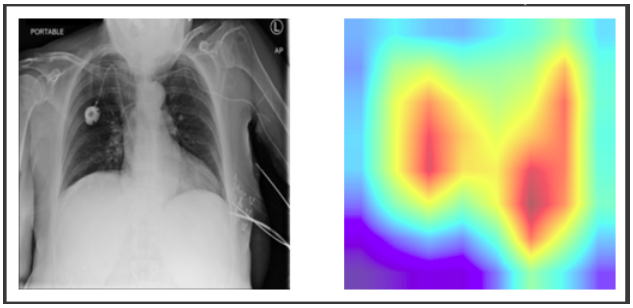

Figura 25. Mapa de Calor de Lung Opacity

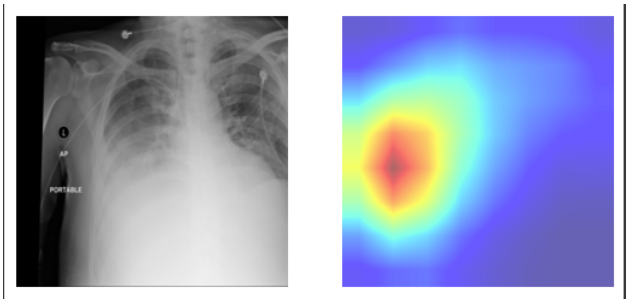

Figura 26. Mapa de Calor de Lung Opacity

VII-C. *Normal*

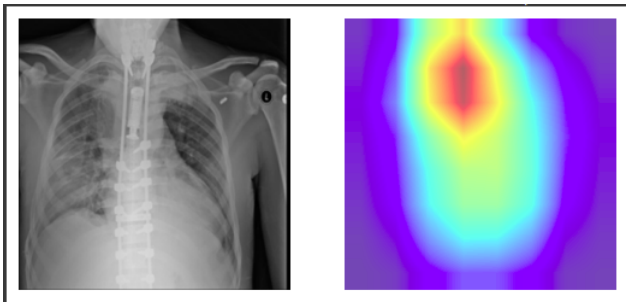

Figura 27. Mapa de Calor de Normal

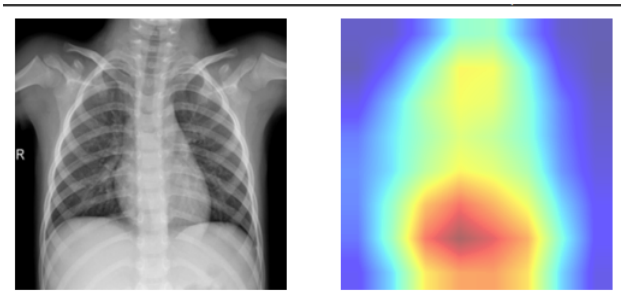

Figura 28. Mapa de Calor de Normal

VII-D. *Viral Pneumonia*

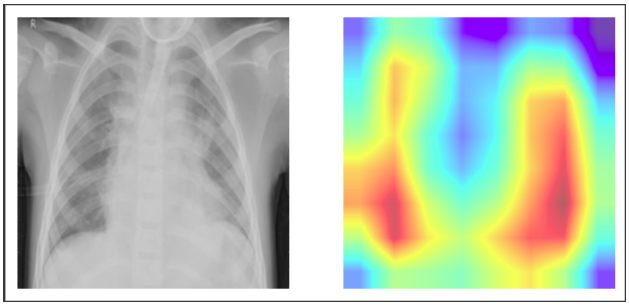

Figura 29. Mapa de Calor de Viral Pneumonia

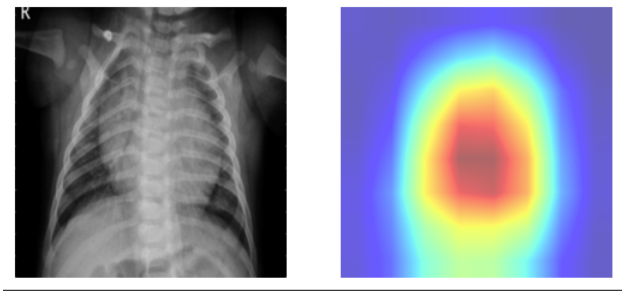

Figura 30. Mapa de Calor de Viral Pneumonia

VIII. Conclusiones

Al realizar el ciclo completo de diseño e implementación de las diferentes redes neuronales, se puede concluir que para las CNN en particular fue posible observar la diferencia en el rendimiento de los modelos al aplicar el filtro sobre el conjunto de entrada, ya que se determinó que el filtro complica la predicción. Por otro lado se obtuvo que para los MLP el cambio no fue tan notorio.

Al comparar los modelos, se logró analizar que los CNN lograron mejores resultados en cuanto a métricas de manera general respecto a las MLP. Adicionalmente, esto lo lograron con tiempos de entrenamiento mucho mayores, en el orden de las horas, mientras que para las MLP de minutos. Por estos motivos se concluye que los CNN implementados resultaron mejores modelos en este caso de estudio particular.

En cuanto a arquitecturas de MLP, se logró concluir que las de mayor cantidad de capas ocultas con reducido número de neuronas fueron las mejores.

Finalmente, al visualizar los mapas de calor tras el entrenamiento de las redes, se lograron determinar las zonas de las imágenes del conjunto de datos que más influenciaron en la toma de decisiones de los modelos. Se concluye que se obtuvieron resultados con sentido, ya que las zonas de mayor calor corresponden al sector de los pulmones para los casos donde se presenta una enfermedad, mientras que para el caso sano se centra en una zona diferente.

Referencias

[1] David E Rumelhart, Geoffrey E Hinton y Ronald J Williams. «Learning representations by back-propagating errors». En: *Nature* 323.6088 (1986), págs. 533-536. URL: http://www.cs.toronto.edu/~hinton/absps/naturebp.pdf.

[2] Yoshua Bengio, Aaron Courville y Pascal Vincent. «Representation learning: A review and new perspectives». En: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.8 (2013), págs. 1798-1828. URL: https://arxiv.org/pdf/1206.5538.pdf.

[3] Alex Krizhevsky, Ilya Sutskhttps://dl.acm.org/doi/pdf/10.1145/3065386ever y Geoffrey E Hinton. «ImageNet classification with deep convolutional neural networks». En: (2012), págs. 1097-1105. URL: https://dl.acm.org/doi/pdf/10.1145/3065386.

[4] Geoffrey E Hinton et al. «Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups». En: *IEEE Signal Processing Magazine* 29.6 (2012), págs. 82-97. URL: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/HintonDengYuEtAl-SPM2012.pdf.

[5] Jacob Devlin et al. «BERT: Pre-training of deep bidirectional transformers for language understanding». En: (2019), págs. 4171-4186. URL: https://arxiv.org/pdf/1810.04805.pdf&usg=ALkJrhhzxlCL6yTht2BRmH9atgvKFxHsxQ.

[6] I. D. Apostolopoulos y T. A. Mpesiana. «Deep Learning for COVID-19 Diagnosis and Prognosis: A Survey». En: *IEEE Journal of Biomedical and Health Informatics* 24.10 (2020), págs. 3028-3048. DOI: 10.1109/JBHI.2020.3014660.

[7] P. K. Sethy y S. K. Behera. «COVID-19 Diagnosis Using Deep Learning with Chest X-ray Images». En: *Biomedical Signal Processing and Control* 64 (2020). DOI: https://doi.org/10.1016/j.compbiomed.2021.104771.

[8] E. E. D. Hemdan, M. A. Shouman y M. E. Karar. «Deep Learning-based Detection for COVID-19 from Chest X-ray Images». En: *Journal of Biomolecular Structure and Dynamics* 39.9 (2020), págs. 3225-3235. DOI: https://doi.org/10.1016/j.procs.2020.10.088.

[9] M. S. Hossain y G. Muhammad. «A Comparative Study on Chest X-Ray Image Classification Using AlexNet, VGGNet, and ResNet». En: *Journal of Healthcare Engineering* 2020 (2020), págs. 1-11. DOI: 10.1155/2020/8885307.