

Proyecto 1 Redes Neuronales



Juan Andrés Contreras

Neyl Peñuela

Juliana Rubio

Facultad de Ingeniería, Pontificia Universidad Javeriana

Técnicas de Aprendizaje de Máquina

Cristian Diaz

2025

Tabla de Contenido

Introducción	2
--------------------	---

Análisis Exploratorio de Datos	3
Clasificación “Normal”	4
Clasificación “Neumonía”	6
Preprocesamiento de Imágenes	7
Diseño de la Red Neuronal	9
Función de Activación y Pérdida	11
Entrenamiento y Evaluación del Modelo	11
Entrenamiento del Modelo	11
Estrategias de Optimización	11
Proceso	12
Evaluación del Modelo	12
Una vez finalizado el entrenamiento, se evaluó el desempeño del modelo mediante el análisis de la precisión y la pérdida tanto en el conjunto de entrenamiento como en el de validación	12
Entrenamiento	12
Prueba	13
Referencias	13

Introducción

Las enfermedades pulmonares, como la neumonía, es una condición de salud crítica que pueden llevar a complicaciones graves si no se diagnostican y tratan a tiempo. Las radiografías de tórax son una herramienta comúnmente utilizada para la detección y el diagnóstico de estas enfermedades, pero la interpretación manual de estas imágenes puede ser un proceso subjetivo y propenso a errores, especialmente en entornos con alta demanda o falta de especialistas. El desarrollo de un sistema automatizado de clasificación de imágenes de rayos X de tórax que pueda identificar de manera confiable la presencia de enfermedades pulmonares podría mejorar significativamente la rapidez y la precisión del diagnóstico, permitiendo una intervención médica más oportuna.

El objetivo es entrenar un modelo de aprendizaje profundo, específicamente una red neuronal convolucional (CNN), que pueda clasificar imágenes de rayos X de tórax y detectar la presencia de neumonía. El proyecto se centrará en aplicar técnicas de

preprocesamiento de imágenes y en diseñar una arquitectura de red que sea efectiva para esta tarea.

Análisis Exploratorio de Datos

Con el fin de analizar las imágenes, se aplicaron diversos filtros para identificar diferenciaciones y límites entre sus componentes, así como patrones relevantes para la comprensión del problema. Y así poder tener un acercamiento al cómo poder pasarle la información a las redes neuronales para que puedan aprender a inferir sobre el problema. Para evitar que estos hallazgos se caractericen de manera subjetiva al ver las imágenes, se decidieron representar estos hallazgos en mediciones con ayuda de la librería `scikit-image` de python. Para ello se decidió tener en cuenta las siguientes mediciones ante la imagen a la hora de su estudio:

Característica	Descripción	Interpretación
Contraste (contrast)	Mide la intensidad de los cambios entre píxeles vecinos.	Valores altos indican texturas con bordes pronunciados y cambios bruscos. Valores bajos sugieren una textura más uniforme.
Disimilitud (dissimilarity)	Similar al contraste, pero menos sensible a valores extremos.	Valores altos indican mayor variabilidad en la imagen. Valores bajos indican una textura más homogénea.
Homogeneidad (homogeneity)	Mide qué tan homogénea es la textura	Valores altos indican que los píxeles vecinos tienen valores similares. Valores bajos sugieren transiciones bruscas.
Energía (energy)	Representa la uniformidad de la textura. Se calcula como la suma de los cuadrados de los valores de la GLCM (Coocurrencia de Niveles de Gris).	Valores altos indican texturas suaves y homogéneas. Valores bajos sugieren una textura más compleja o con ruido.
Correlación (correlation)	Mide la relación entre los valores de píxeles en la imagen.	Valores cercanos a 1 indican una textura estructurada con patrones claros. Valores bajos sugieren aleatoriedad.

Tabla (1): Métricas de estudio.

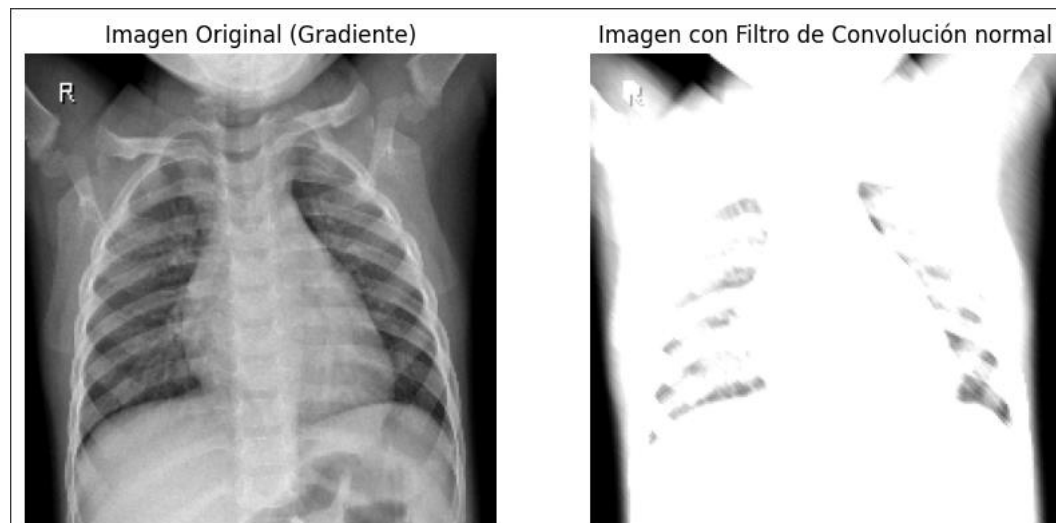
En el estudio, se decidió dividir el estudio entre dos categorías si el pulmón está “Normal” o se encuentra afectado por neumonía.

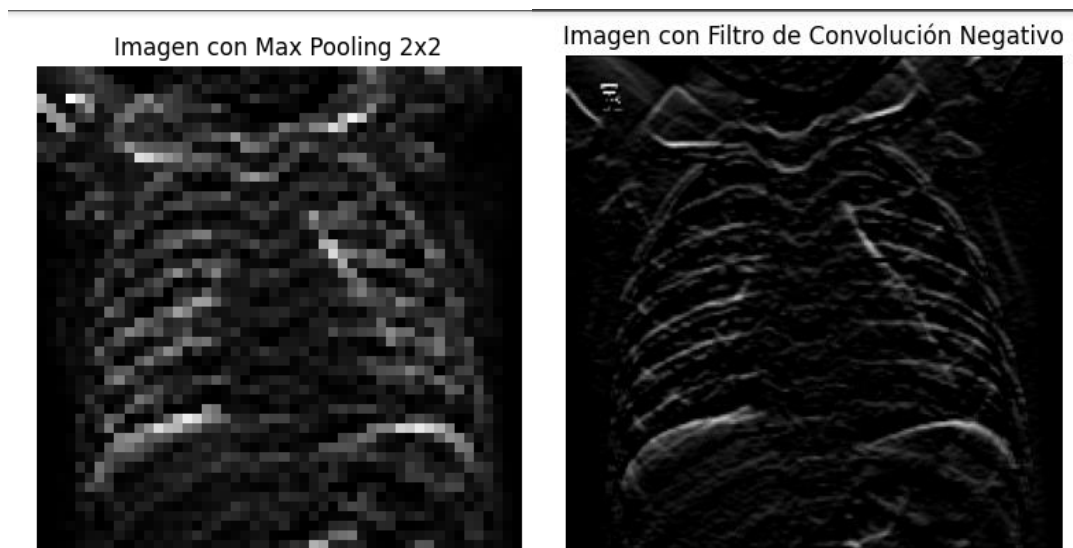
Como primer estudio se tomó el primer filtro el cual fue generar una convolución con una matriz de identidad (I) para poder potenciar los límites, tal como lo haría una “máscara”, bajo la idea de intentar encontrar alguna diferenciación específica al potenciar las texturas y la intensidad de las formas.

Por otro lado, buscando otra perspectiva, a la imagen original se le aplicó un filtro negativo, con la razón de poder encontrar algún tipo de *homogeneidad*, ya que esta se “caracteriza a la consolidación neumónica. La neumonía bacteriana se comporta radiográficamente como una opacidad homogénea, debido a la confluencia de acinos consolidados. “(Estevan, n.d.).

El tercer filtro, el tercer estudio, se fijó más en encontrar alguna distribución no segmentaria, puesto que, “la infección en la neumonía bacteriana, por su forma de propagación, no respeta los límites entre los segmentos (distribución no segmentaria).” (Estevan, n.d.). Para ello se decidió aplicar una operación Max Pooling ya que, en este proceso las características más robustas y generales, en si simplificando las formas y texturas ayudando a ver la distribución de la enfermedad.

Clasificación “Normal”

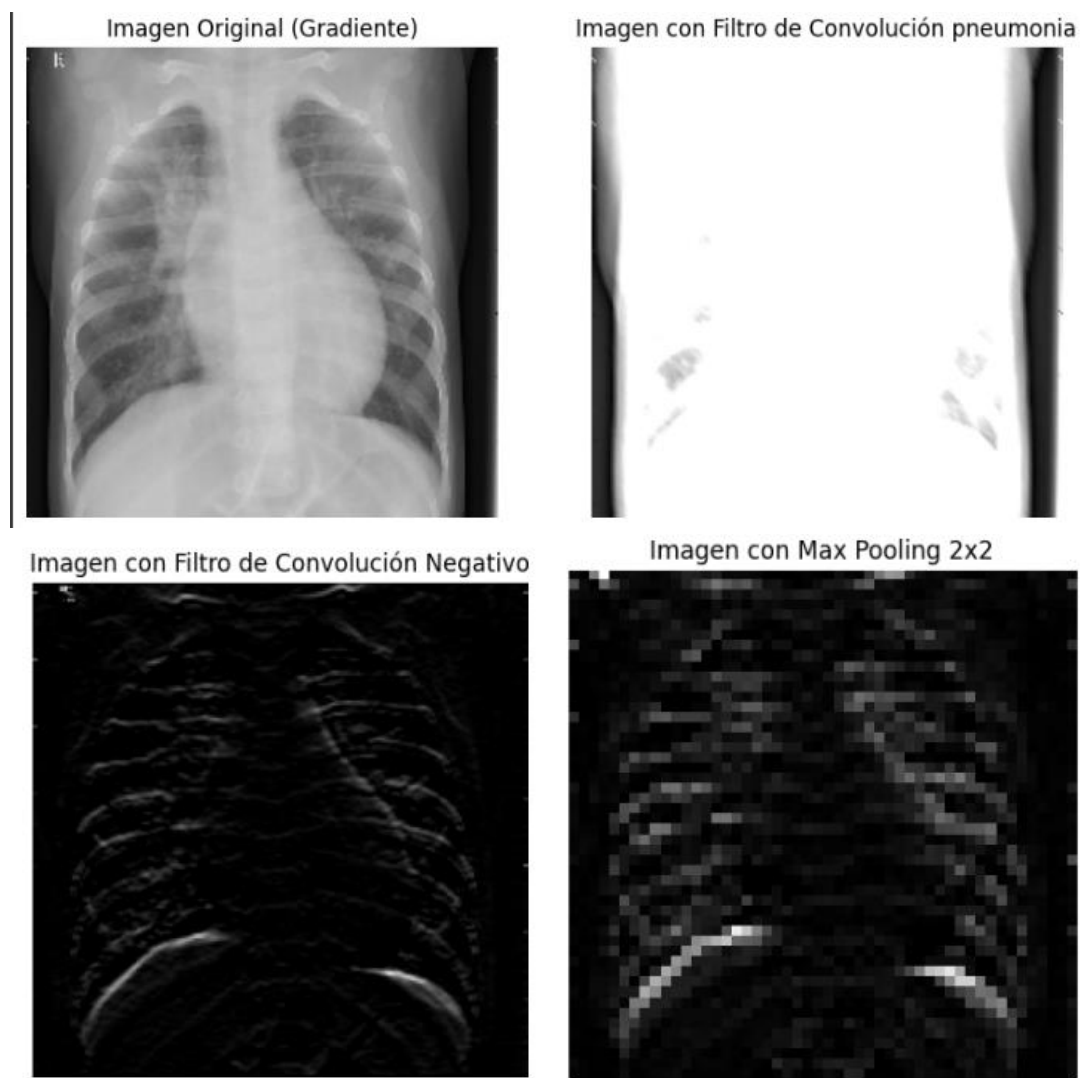




Característica	Imagen Convolucionada	Imagen Original	Filtro Negativo	Imagen Pooling
Contraste	111.52	176.62	352.54	807.99
Disimilitud	3.34	7.82	9.42	18.08
Homogeneidad	0.82	0.19	0.43	0.13
Energía	0.76	0.06	0.37	0.05
Correlación	0.99	0.97	0.65	0.4

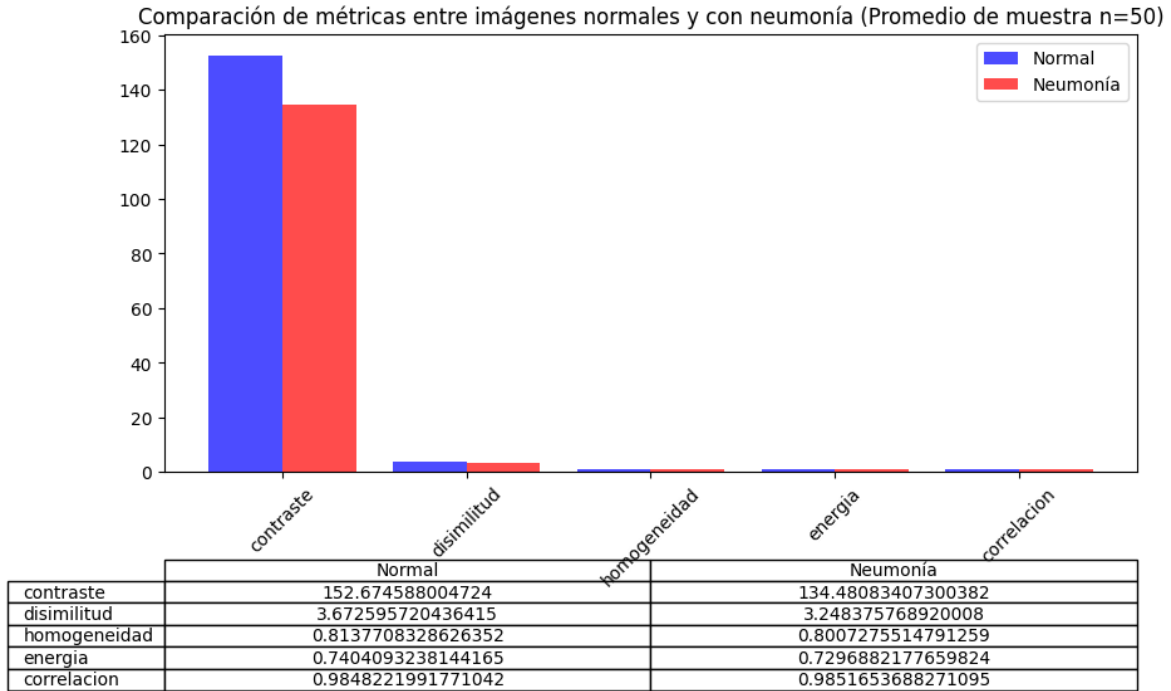
Tabla (2): Resultados imagen clasificada como Normal.

Clasificación “Neumonía”



Característica	Imagen Convolucionada	Imagen Original	Filtro Negativo	Imagen Pooling
Contraste	96.12	91.4	119.02	348.21
Disimilitud	2.86	5.13	4.92	10.52
Homogeneidad	0.83	0.27	0.49	0.19
Energía	0.8	0.03	0.37	0.06
Correlación	0.99	0.99	0.7	0.42

Tabla (3): Resultados imagen clasificada como Neumonia.



Los resultados sugieren que las imágenes de neumonía exhiben una mayor dispersión de patrones y rugosidades complejas en comparación con pulmones sanos. Esta información podría ser crucial para identificar bronquiolos respiratorios (BR), conductos alveolares (CA), sacos alveolares (SA) y alvéolos afectados por la neumonía.

Preprocesamiento de Imágenes

Teniendo en cuenta lo anterior, se propone usar capas de Conv2D en Keras, la metodología con la cual opera es la aplicación de filtros (también conocidos como kernels) Estos filtros son pequeños arreglos de números diseñados para detectar características específicas, tales como bordes, esquinas, texturas y patrones. Tal como se veía propuesto y necesario con el punto anterior.

Entonces, ¿por qué es necesario para el proceso?, porque su centro es la convolución: el filtro se desliza a través de la imagen, realizando una operación matemática en cada posición. Esta operación implica la multiplicación de los valores del filtro por los valores correspondientes en la imagen, seguida de la suma de estos productos. El resultado de esta operación se almacena en un mapa de características, que resalta las áreas de la imagen donde se detecta la característica correspondiente al filtro.

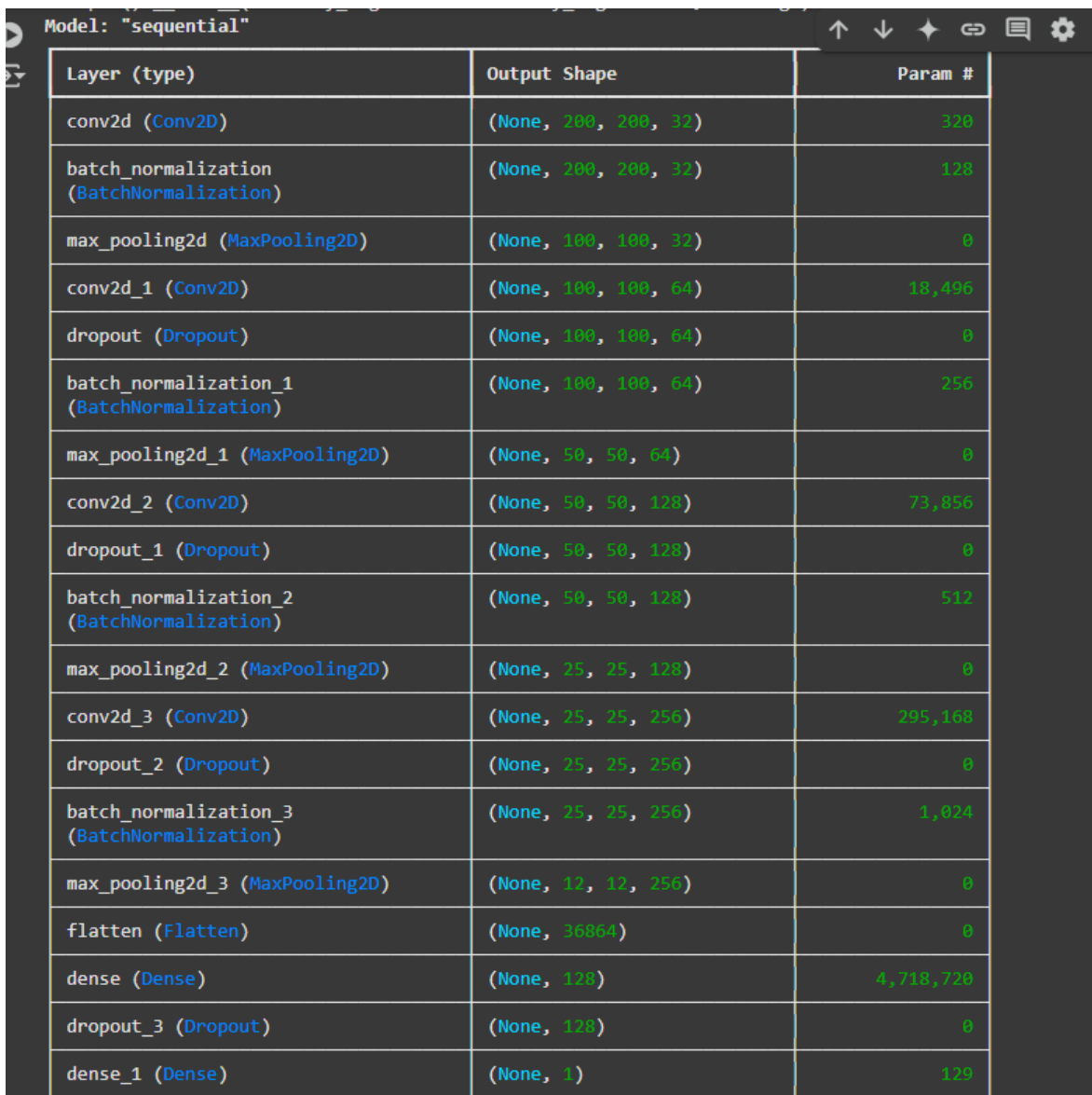
Cada filtro empleado genera su propio mapa de características, lo que permite a la capa Conv2D capturar múltiples patrones en la imagen de entrada, toma filtros negativos, filtros

convolucionales, “Max pooling” y muchos más filtros que ayudan a desglosar las imágenes. Además, a medida que la información avanza a través de las capas de una CNN, las primeras capas tienden a detectar características simples, mientras que las capas posteriores combinan estas características para identificar patrones más complejos.

Agregando a lo anterior, esto no significa que la capa hará todo el trabajo, se necesita de normalizar los datos primero; para normalizar las imágenes se deben dividir los valores de píxeles por 255. Esta normalización reduce el rango de los datos de entrada, lo que facilita la convergencia del gradiente descendente durante el entrenamiento y, por lo tanto, ayuda a minimizar la función de pérdida y a reducir la tasa de error del modelo.

Así mismo, se decidió agregar aleatoriedad a las imágenes para que el modelo sea más resiliente, generando un aumento de datos. El cual se aplicó un aumento de datos utilizando ImageDataGenerator de Keras. Este proceso incluyó rotaciones aleatorias de hasta 20 grados, zooms aleatorios de hasta un 15%, desplazamientos horizontales y verticales aleatorios de un 10%, y volteo horizontal aleatorio. Estas transformaciones aumentaron la diversidad del conjunto de datos de entrenamiento, lo que mejoró la capacidad del modelo para generalizar y, por lo tanto, ayudó a minimizar la función de pérdida y a reducir la tasa de error.

Diseño de la Red Neuronal



The image shows a screenshot of a Keras model summary for a sequential model. The model is named "sequential". The summary table lists 18 layers, their types, output shapes, and the number of parameters. The layers are: conv2d (Conv2D), batch_normalization (BatchNormalization), max_pooling2d (MaxPooling2D), conv2d_1 (Conv2D), dropout (Dropout), batch_normalization_1 (BatchNormalization), max_pooling2d_1 (MaxPooling2D), conv2d_2 (Conv2D), dropout_1 (Dropout), batch_normalization_2 (BatchNormalization), max_pooling2d_2 (MaxPooling2D), conv2d_3 (Conv2D), dropout_2 (Dropout), batch_normalization_3 (BatchNormalization), max_pooling2d_3 (MaxPooling2D), flatten (Flatten), dense (Dense), dropout_3 (Dropout), and dense_1 (Dense). The total number of parameters is 295,168.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 200, 200, 32)	320
batch_normalization (BatchNormalization)	(None, 200, 200, 32)	128
max_pooling2d (MaxPooling2D)	(None, 100, 100, 32)	0
conv2d_1 (Conv2D)	(None, 100, 100, 64)	18,496
dropout (Dropout)	(None, 100, 100, 64)	0
batch_normalization_1 (BatchNormalization)	(None, 100, 100, 64)	256
max_pooling2d_1 (MaxPooling2D)	(None, 50, 50, 64)	0
conv2d_2 (Conv2D)	(None, 50, 50, 128)	73,856
dropout_1 (Dropout)	(None, 50, 50, 128)	0
batch_normalization_2 (BatchNormalization)	(None, 50, 50, 128)	512
max_pooling2d_2 (MaxPooling2D)	(None, 25, 25, 128)	0
conv2d_3 (Conv2D)	(None, 25, 25, 256)	295,168
dropout_2 (Dropout)	(None, 25, 25, 256)	0
batch_normalization_3 (BatchNormalization)	(None, 25, 25, 256)	1,024
max_pooling2d_3 (MaxPooling2D)	(None, 12, 12, 256)	0
flatten (Flatten)	(None, 36864)	0
dense (Dense)	(None, 128)	4,718,720
dropout_3 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 1)	129

Arquitectura de la Red

Para detectar neumonía en imágenes de rayos X, diseñamos una red neuronal convolucional (CNN). Esta red está compuesta por varias capas: convolucionales, de normalización, de agrupamiento (pooling) y densas. La idea detrás de esta estructura es que las capas convolucionales extraen características importantes de las imágenes, como bordes o texturas, mientras que las capas de pooling reducen el tamaño de los datos, manteniendo la información más relevante. Finalmente, las capas densas se encargan de clasificar las imágenes, permitiendo que el modelo distinga entre casos normales y casos de neumonía de manera eficiente.

Componentes

Capas Convolucionales: Usamos cuatro bloques convolucionales con filtros de 32, 64, 128 y 256, respectivamente. Estas capas son clave para extraer características y patrones espaciales de las imágenes, permitiendo que el modelo detecte detalles importantes a diferentes niveles de complejidad.

Batch Normalization: Después de cada convolución, aplicamos normalización por lotes. Esto ayuda a estabilizar el entrenamiento, acelera la convergencia del modelo y reduce la dependencia de una buena inicialización de los pesos.

Capas de Agrupamiento (MaxPooling): Incluimos capas de MaxPooling para reducir el tamaño de las imágenes, lo que mejora la eficiencia computacional y conserva la información más relevante, descartando detalles menos importantes.

Dropout: Para evitar que el modelo se sobreajuste a los datos de entrenamiento, añadimos capas de Dropout. Estas desactivan aleatoriamente algunas neuronas durante el entrenamiento, lo que obliga al modelo a aprender de manera más robusta y generalizable.

Capas Densas: Finalmente, conectamos una capa densa de 128 neuronas con activación ReLU, seguida de una capa de salida con activación sigmoide. Esta última es la encargada de realizar la clasificación binaria, determinando si una imagen es normal o presenta neumonía.

Función de Activación y Pérdida

Función de Activación en la Capa de Salida (Sigmoide):

Se usa la función de activación sigmoide porque el modelo está diseñado para clasificación binaria. Esta función transforma la salida en un valor entre 0 y 1, que podemos interpretar como la probabilidad de que la entrada pertenezca a la clase positiva (en este caso, neumonía).

Por otro lado, cuando se trata de problemas de clasificación con más de dos clases (clasificación multiclase), se utiliza la función Softmax. Esta función asigna probabilidades a cada una de las clases, asegurándose de que la suma de todas las probabilidades sea igual a 1. Así, el modelo puede determinar la clase más probable de manera efectiva • Función de

Pérdida (Binary Crossentropy):

Se utiliza la entropía cruzada binaria porque estamos trabajando con un problema de clasificación binaria. Esta función mide la diferencia entre la predicción del modelo y la etiqueta real, y tiene la particularidad de penalizar más los errores en la clasificación. Es decir, cuanto más se aleje la predicción del valor real, mayor será la penalización, lo que ayuda a que el modelo aprenda de manera más eficiente.

Entrenamiento y Evaluación del Modelo

Entrenamiento del Modelo

Para entrenar el modelo CNN en la clasificación de imágenes de rayos X como normales o con neumonía, se aplicaron técnicas de optimización y prevención del sobreajuste. Esto resultó en un mejor desempeño en la detección de neumonía en radiografías, con menor riesgo de sobreajuste y mayor capacidad de generalización.

Estrategias de Optimización

Se emplearon dos estrategias clave:

1. Reducción de la Tasa de Aprendizaje (ReduceLROnPlateau):

Disminuye la tasa de aprendizaje cuando la pérdida en validación deja de mejorar. (Keras, 2025)

2. Detención Temprana (EarlyStopping):

Finaliza el entrenamiento cuando la pérdida en validación no mejora después de varias épocas. (Keras, 2025)

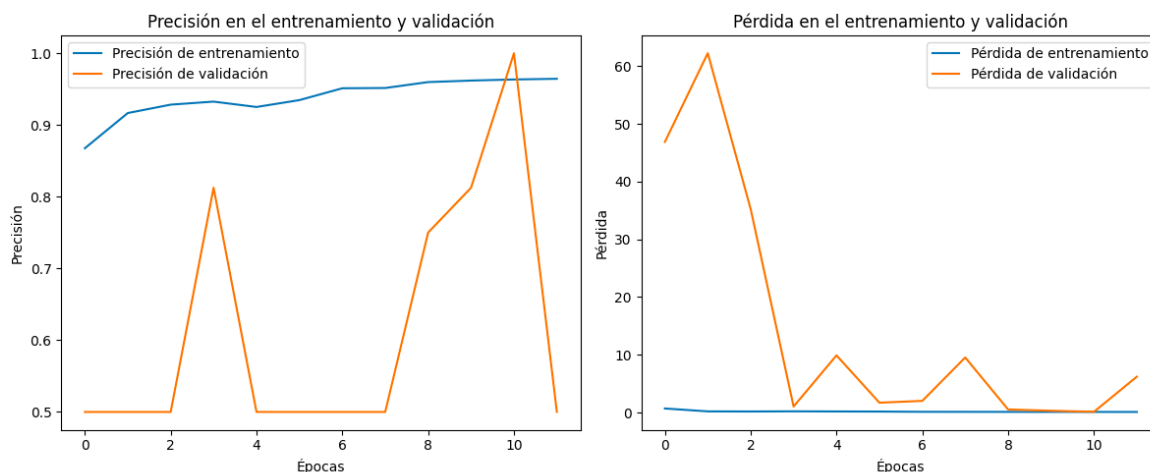
Proceso

El modelo fue entrenado utilizando `datagen.flow` que aplica técnicas de aumento de datos para mejorar la generalización. Se utilizó un tamaño de lote de 32 y validación en cada época durante 12 iteraciones:

Evaluación del Modelo

Una vez finalizado el entrenamiento, se evaluó el desempeño del modelo mediante el análisis de la precisión y la pérdida tanto en el conjunto de entrenamiento como en el de validación

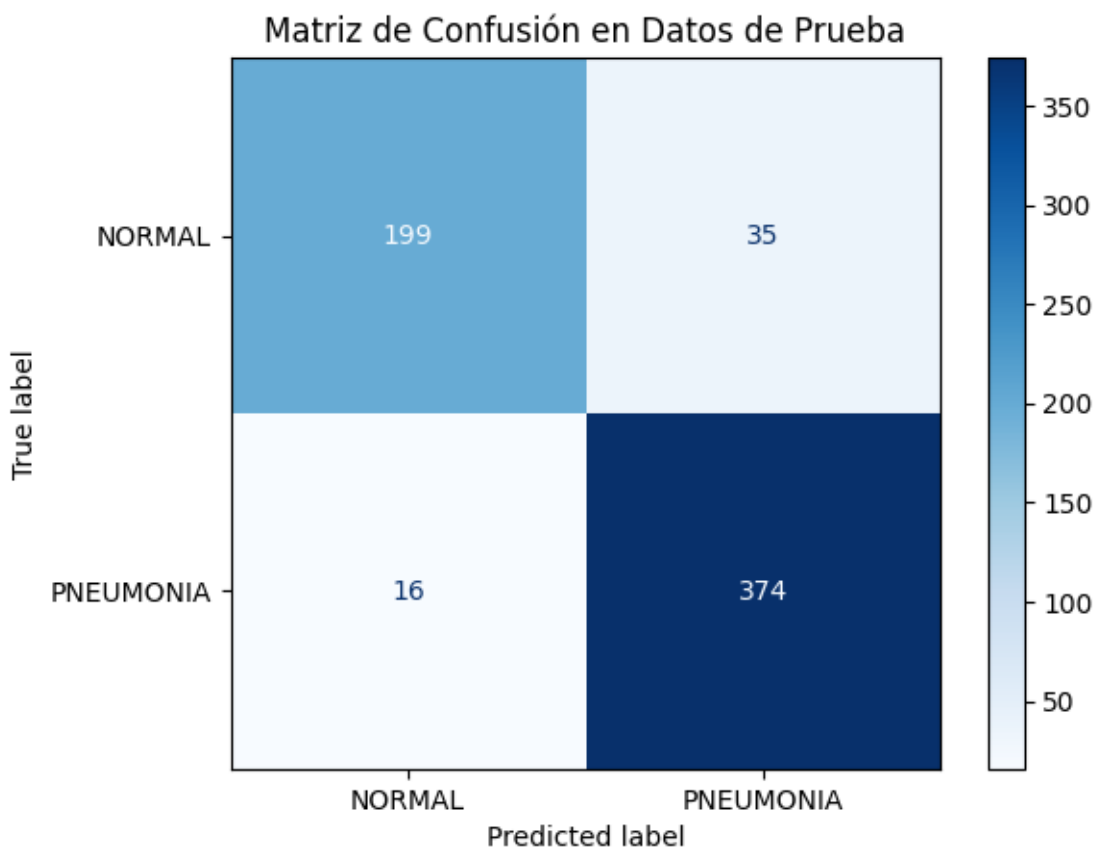
Entrenamiento



- **Precisión:** El modelo muestra una mejora progresiva en la precisión del entrenamiento y validación, indicando un aprendizaje efectivo.
- **Pérdida:** La pérdida en entrenamiento y validación sigue una tendencia decreciente, lo que sugiere que el modelo es adecuado.

Prueba

Para medir el desempeño del modelo en un conjunto de datos no visto durante el entrenamiento, se generó una matriz de confusión y un reporte de métricas de clasificación:



- **Matriz de Confusión:**

- El modelo demuestra un desempeño sólido en la detección de neumonía y casos normales.
- El modelo clasifica correctamente 199 imágenes normales y 374 imágenes con neumonía.
- La cantidad de falsos negativos ha disminuido notablemente (solo 16 casos de neumonía mal clasificados como normales).
- Se identifican 35 falsos positivos en la clase normal.

	precision	recall	f1-score	support
NORMAL	0.93	0.85	0.89	234
PNEUMONIA	0.91	0.96	0.94	390
accuracy			0.92	624
macro avg	0.92	0.90	0.91	624
weighted avg	0.92	0.92	0.92	624
Precisión global del modelo: 0.92				

Precisión:

Normal (93%): Esto significa que, de todas las imágenes que el modelo clasificó como normales, el 93% realmente eran normales. En otras palabras, hay un 7% de falsos positivos (imágenes que el modelo pensó que eran normales, pero en realidad no lo eran).

Neumonía (91%): Aquí, el 91% de las imágenes clasificadas como neumonía corresponden a casos reales de neumonía. El 9% restante son falsos positivos (imágenes que el modelo confundió con neumonía, pero que en realidad no la tenían).

Recall (Sensibilidad o Tasa de Verdaderos Positivos):

Normal (85%): El modelo identificó correctamente el 85% de las imágenes normales, pero se equivocó en el 15% de los casos (falsos negativos).

Neumonía (96%): Aquí, el modelo detectó correctamente el 96% de los casos de neumonía, lo que significa que solo se le escapó el 4% de los casos (falsos negativos).

F1-Score:

Normal (0.89): Esta métrica combina precisión y recall, y un valor de 0.89 indica que el modelo tiene un buen equilibrio entre ambas para la clase normal.

Neumonía (0.94): Un F1-Score de 0.94 sugiere que el modelo es muy efectivo para detectar neumonía, manteniendo un balance entre precisión y recall.

Precisión Global (Accuracy - 92%):

Esto significa que, en general, el modelo clasificó correctamente el 92% de todas las imágenes de prueba. Es una métrica global que refleja cuántas predicciones fueron acertadas en total.

Conclusiones

Desafíos Encontrados

- **Desbalance de Clases:** Nos dimos cuenta de que el conjunto de datos tenía muchas más imágenes de neumonía que de casos normales. Esto podía sesgar el modelo, haciendo que aprendiera a favorecer una clase sobre la otra.
- **Sobreajuste del Modelo:** Al principio, el modelo se desempeñaba muy bien con los datos de entrenamiento, pero su rendimiento caía con los datos de validación. Esto era una señal clara de que el modelo estaba memorizando los datos en lugar de generalizar.
- **Tiempo de Entrenamiento:** Debido a la complejidad del modelo, el entrenamiento tomaba mucho tiempo, lo que limitaba nuestra capacidad para probar diferentes configuraciones y ajustar los hiperparámetros de manera eficiente.

Decisiones Tomadas

- **Aumento de Datos:** Para abordar el desbalance, aplicamos técnicas de aumento de datos, como rotaciones, zoom y desplazamientos. Esto nos permitió generar más variaciones de las imágenes existentes, ayudando al modelo a generalizar mejor.
- **Dropout y Batch Normalization:** Para combatir el sobreajuste, añadimos capas de dropout y Batch normalization. Estas técnicas ayudaron a estabilizar el entrenamiento y a mejorar el rendimiento del modelo con datos nuevos.
- **ReduceLROnPlateau y Early Stopping:** Implementamos ajustes dinámicos en la tasa de aprendizaje y usamos detención temprana. Esto nos ayudó a evitar entrenamientos innecesarios y a asegurar que el modelo convergiera de manera más eficiente.

Mejoras Futuras

- **Ampliar el Conjunto de Datos:** Sería ideal incluir más imágenes de diferentes fuentes y calidades. Esto ayudaría a que el modelo sea más robusto y capaz de manejar una mayor variedad de casos.
- **Optimizar la Arquitectura:** Podríamos experimentar ajustando el número de filtros en las capas convolucionales o probando funciones de activación alternativas en las capas densas, como RIDGE o LASSO.
- **Evaluar con Métricas Adicionales:** Además de las métricas actuales, podríamos usar el AUC-ROC para tener una mejor comprensión de la capacidad del modelo para distinguir entre casos normales y de neumonía.

Referencias

- Estevan, M. (n.d.). *Examen radiográfico del tórax en las neumonías de probable causa bacteriana.*

http://www.scielo.edu.uy/scielo.php?script=sci_arttext&pid=S1688-12492002000100004

- Keras. (2025). *Keras documentation: ReduceLROnPlateau*. Keras.io.
https://keras.io/api/callbacks/reduce_lr_on_plateau/
- Keras, K. (2025). *Keras documentation: EarlyStopping*. Keras.io.
https://keras.io/api/callbacks/early_stopping/