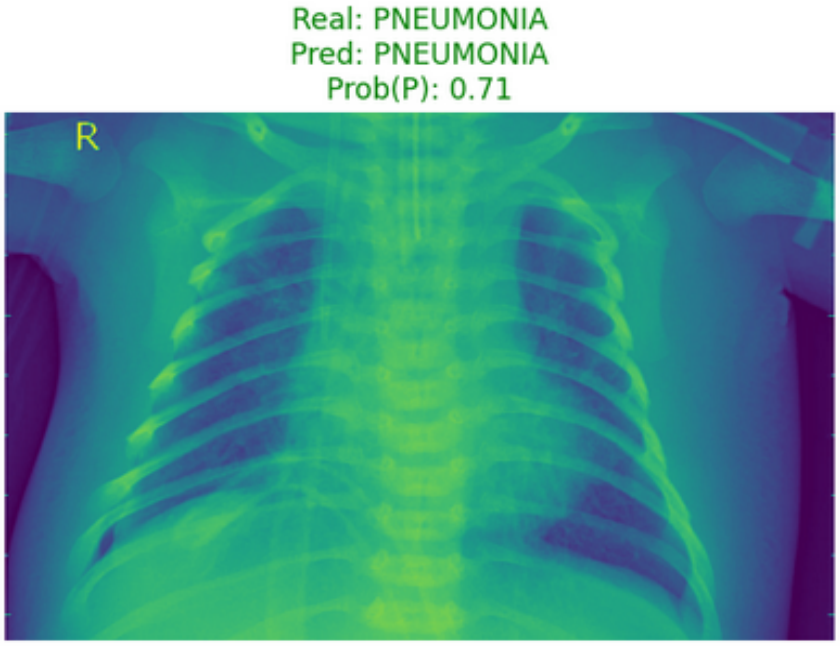
Informe de Desarrollo: Modelo de Detección de Neumonía por Radiografía



Proyecto para el Instituto Superior Politécnico de Córdoba
Procesamiento de imágenes y Modelos de IA.
Profesores: Sol Figueroa y Carlos Charletti.

1. Introducción y Objetivo del Proyecto

El objetivo principal de este proyecto fue desarrollar un modelo de Deep Learning capaz de clasificar imágenes de radiografías de tórax para identificar la presencia de neumonía. Se buscaba crear una herramienta de apoyo al diagnóstico que mejorará la eficiencia y precisión en la detección de esta condición. Para ello, se trabajó con un dataset de imágenes de radiografías de tórax (clases "NORMAL" y "PNEUMONIA"), divididas en conjuntos de entrenamiento, validación y prueba.

dataset: https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia

2. Preparación y Aumento de Datos (Data Augmentation)

Desde el inicio del proyecto, se implementó una estrategia robusta de **aumento de datos** para mitigar el sobreajuste y mejorar la capacidad de generalización del modelo, dado el tamaño limitado del dataset. Todas las imágenes fueron reescaladas a **150x150 píxeles** y normalizadas a un rango de píxeles [0,1].

Las técnicas de aumento de datos aplicadas incluyeron:

- Reescalado: 1./255
- Rotación: rotation_range=20
- Desplazamiento: width_shift_range=0.2, height_shift_range=0.2
- Cizallamiento: shear_range=0.25
- Zoom: zoom_range=0.25
- Ajuste de Brillo: brightness_range=[0.7, 1.3]
- Ajuste de Canales: channel_shift_range=75
- Modo de Relleno: fill_mode='nearest'

3. Iteraciones de Modelado y Resultados Detallados

A lo largo del desarrollo, el principal desafío fue combatir un **sobreajuste severo y persistente**, manifestado por un **sesgo de predicción única** (el modelo tendía a predecir siempre una misma clase) en el conjunto de validación. A continuación, se detallan las distintas iteraciones de diseño y ajuste del modelo, junto con sus resultados:

Intento Nº1: CNN desde Cero - Configuración Inicial

Arquitectura: Una Red Neuronal Convolucional (CNN) construida desde cero. Se iniciaron las pruebas con una configuración base de capas convolucionales y de pooling.

Comportamiento Observado:

- Las etiquetas reales para las imágenes normales eran 0, pero las predicciones de probabilidad eran muy altas (~0.97 - 0.99).
- Al aplicar un umbral de 0.5, todas las predicciones resultaron en 1 (Neumonía).
- Pérdida en Test: 2.6550 (muy alta), indicando una nula capacidad de generalización.
- Precisión en Test: 0.5000, reflejando que el modelo predecía todo como "Neumonía" y, dado un dataset balanceado, solo acertaba el 50% de las veces.

Conclusión: El modelo no estaba aprendiendo a distinguir entre las clases; simplemente predecía la clase "Neumonía" de forma indiscriminada.

Intento Nº2: CNN desde Cero - Aumento de Capacidad (Más Filtros)

Acción Tomada: Se incrementó la capacidad de la red **añadiendo más filtros** a las capas convolucionales (ej. de 32-64-128 a **64-128-256**). La hipótesis era darle al modelo más "neuronas" y herramientas para aprender patrones más complejos.

Comportamiento Observado:

- loss en entrenamiento: Disminuía constantemente (de 2.9182 a 1.2375), lo que confirmaba que el modelo memorizaba los datos de entrenamiento.
- accuracy en entrenamiento: Aumentaba significativamente (de 0.7227 a 0.8403).
- val_accuracy (validación): Se estancó en 0.5052 (aproximadamente 0.50), sin mejora real.
- val_loss (validación): Se disparó enormemente (de 1.3703 a 24.3714), la señal más clara de sobreajuste.
- Evaluación en Test: loss: 2.3092, accuracy: 0.6218. Aunque la precisión en test mejoró ligeramente, la pérdida seguía siendo altísima, indicando una generalización deficiente.

Conclusión: El modelo seguía prediciendo una sola clase en validación/test, y el aumento de filtros solo facilitó que memorizara en lugar de generalizar.

Intento Nº3: CNN desde Cero - Reducción del Learning Rate y Adición de Early Stopping

Acción Tomada: Ante el sobreajuste severo, se redujo el learning rate de 0.0001 a un valor muy bajo (0.000005). La intención era permitir que el optimizador realizara pasos más pequeños y estables. También se añadió un callback EarlyStopping para monitorear la val_loss y detener el entrenamiento cuando no hubiera mejoras significativas, restaurando los mejores pesos del modelo.

Comportamiento Observado:

- loss y val_loss: Ambos disminuyeron y se mantuvieron muy cerca (ej. 0.0970 vs 0.0943), lo cual era bueno en términos de evitar el sobreajuste.
- accuracy en entrenamiento: Muy baja y errática (de 0.4013 a 0.4727).
- val_accuracy: Se mantuvo estancada en 0.4948 (aproximadamente 0.50).
- Predicciones Específicas: Para imágenes NORMAL, el modelo predecía NORMAL (probabilidades ~0.3-0.4). Para imágenes PNEUMONIA, el modelo también predecía NORMAL (probabilidades ~0.3-0.4)!
- Evaluación en Test: loss: 0.0943, accuracy: 0.4979. Consistentemente alrededor del 50%.

Conclusión: El learning_rate excesivamente bajo impidió que el modelo aprendiera las características distintivas de la neumonía. La red se quedó "atrapada" en un estado donde siempre predecía "Normal".z

Intento Nº4: CNN desde Cero - Ajuste del Learning Rate (Ligero Aumento) y Regularización Reforzada

Acción Tomada: Se buscó un balance ajustando nuevamente el learning_rate de 0.00001 a 0.00005 (un ligero aumento respecto al intento anterior, pero aún muy bajo). Dropout: Aumentado a 0.6 (o 0.7). kernel_regularizer: Aumentado regularizers.l2(0.005) o 0.01 en las capas Dense.

Comportamiento Observado:

- Métricas de Entrenamiento: loss disminuye, accuracy aumentaba (ej. hasta 0.8514), precision, recall, auc mejoraban. Esto indicaba que el modelo memorizaba los datos de entrenamiento muy bien.
- val_loss (validación): Empieza en 0.9500 y luego aumenta significativamente (ej. a 1.4811).
- val_accuracy y val_precision (validación): Estancadas en 0.5042 - 0.5063.
- val_recall (validación): ¡Se mantuvo en 1.0000 (100%) en todas las épocas! Esto significaba que el modelo acertaba todas las neumonías reales, pero lo hacía clasificando *todas* las imágenes como "Neumonía", incluyendo las normales.
- Evaluación en Test: loss: 0.9538, accuracy: 0.5000, precision: 0.5000, recall: 1.0000. Consistentemente sesgado.

Conclusión: La CNN entrenada desde cero, incluso con aumento de datos y fuertes regularizaciones, no lograba aprender las características discriminatorias de forma generalizable. Tenía demasiada capacidad para la cantidad de datos, lo que la llevaba a memorizar patrones en lugar de entenderlos.

Intento Nº5: Exploración de AveragePooling2D (Alternativa de Pooling)

Acción Tomada: Se consideró y exploró la posibilidad de reemplazar las capas MaxPooling2D con AveragePooling2D. La hipótesis era que AveragePooling2D podría ser más suave, retener más información espacial y, potencialmente, ayudar al modelo a capturar características más sutiles.

Conclusión Previa: Aunque podía ofrecer ventajas teóricas, el problema subyacente de la incapacidad de la CNN desde cero para generalizar con el dataset actual era demasiado grande para ser resuelto solo con un cambio en el tipo de pooling.

Intento Nº6: Últimos cambios (batch size)

- **Acción Tomada:** Se volvió a usar MaxPooling, se bajó los filtros de nuevo a 32, 64, 128 por que al parecer era lo adecuado para la cantidad de datos y para probar algo nuevo bajamos el batch size a 16 Para probar gradientes un poco más "ruidosos" (que a veces ayudan a la convergencia). Serían 480/16=30 actualizaciones de pesos por época.
- **Conclusión Previa:** La raíz del problema sigue siendo la capacidad del modelo en relación con la escasez de datos. Un batch_size más pequeño puede generar gradientes más ruidosos que *podrían* ayudar a escapar de mínimos locales, pero no puede compensar la falta de patrones generalizables en el dataset para una red de esta complejidad.Lo que está sucediendo es que el modelo está aprendiendo muy bien de los datos de entrenamiento(métricas de precision y recall son razonables y es alto el accuary), ¡pero no está generalizando en absoluto a los datos de validación en val loss y val accuary, de nuevo predice solo una clase.

Punto de Quiebre: Agotamiento de Estrategias con CNN desde Cero

El problema fue que el modelo tenía que aprender desde 0 a distinguir formas, reconocer patrones complejos con pocos datos, por lo que solo memorizo las de entrenamiento, teniendo un sobreajuste extremo. Las redes convolucionales están diseñadas para aprender jerarquías de características: las primeras capas aprenden bordes y esquinas, las capas intermedias aprenden texturas y partes de objetos, y las capas profundas aprenden objetos completos.

Aprender esta jerarquía completa desde cero requiere una cantidad masiva de datos (millones de imágenes), que es lo que se usa para entrenar modelos como VGG16.

VGG16 fue entrenado con el dataset de **ImageNet**, que contiene **millones de imágenes** (más de 14 millones) y miles de categorías (1000 clases)..

Intento Nº6: Transfer Learning con VGG16 (Estrategia Exitosa)

Transfer Learning. Esta técnica es ideal para datasets de imágenes de tamaño limitado, ya que aprovecha el "conocimiento" de un modelo pre-entrenado en un dataset masivo (ImageNet) para tareas similares, adaptándolo a la tarea específica con un entrenamiento mínimo.

Modelo Base: Se cargó **VGG16** pre-entrenado en ImageNet

Congelación de Capas: Todas las capas convolucionales del VGG16 fueron congeladas (layer.trainable = False). Esto significó que el VGG16 actuaría como un extractor de características fijas.

Capas de Clasificación Personalizadas (Head): Se añadió un pequeño "head" de clasificación encima del VGG16, compuesto por:

Evaluación Final del Modelo (Conjunto de Prueba): El modelo final, entrenado con la estrategia de Transfer Learning (VGG16 con capas base congeladas), fue evaluado en el conjunto de prueba (test_generator), que contiene imágenes completamente nuevas y no vistas durante el entrenamiento o la validación.

- **Interpretación de Resultados:**
    - **Accuracy:** 0.7671 (76.71%) El modelo clasifica correctamente más de tres cuartas partes de las imágenes en datos no vistos, un gran avance.
    - **Precisión (Neumonía):** Excepcionalmente alta. Cuando el modelo predice "Neumonía", tiene una **alta fiabilidad** (más del 95% de acierto), lo cual es crucial para minimizar falsos diagnósticos.
    - **Recall (Neumonía):** El modelo logra identificar el 55.98% de los casos *reales* de Neumonía. Si bien es un muy buen punto de partida, indica que aún hay margen para mejorar la detección de todos los casos de neumonía y reducir los falsos negativos.
    - **AUC:** Un valor muy cercano a 1.0, (0.9340) lo que confirma una **excelente capacidad discriminatoria** general del modelo para distinguir entre las dos clases ("Normal" y "Neumonía").

Intento Nº7: Implementación de mejoras a Transfer Learning usando Fine Tunning

Luego de tomar el modelo pre-entrenado (VGG16) y **congar todas sus capas convolucionales**. Significando que los pesos de VGG16 no cambiaban durante el entrenamiento.

Se añadieron propias capas de clasificación (el "head" con Flatten, Dense, Dropout).

VGG16 actuaba como un "cerebro" fijo que extraía características generales (bordes, texturas, formas) de las imágenes, y las capas aprendían a clasificar la neumonía basándose en esas características. Demostró que las características generales de VGG16 eran útiles para el problema. Obteniendo una Precisión muy alta (0.9562).

Implementación de Fine Tuning "Descongelar" y Aprender Más

En lugar de mantener *todas* las capas de VGG16 congeladas, este código selecciona una parte de las **últimas capas convolucionales** del modelo base (base_model.layers[-num_layers_to_unfreeze:]) y las marca como **entrenables (layer.trainable = True)**.

num_layers_to_unfreeze = 4 significa que las últimas 4 capas convolucionales de VGG16 ahora podrán **ajustar sus pesos**.

¿Por qué las últimas capas? Las primeras capas de una CNN pre-entrenada aprenden características muy generales (líneas, círculos, texturas básicas) que son universales para cualquier imagen. Las **capas más profundas (las últimas)** aprenden características de más alto nivel y más específicas (como formas de objetos, en este caso, patrones que podrían indicar neumonía). Al "descongelar" estas últimas capas,se le está dando la oportunidad de **adaptarse y especializarse** en los patrones específicos en lugar de solo usar los patrones generales que aprendieron de ImageNet (que son fotos de gatos, coches, etc.).

1. **Re-compilar el Modelo:**
   Después de cambiar la propiedad trainable de las capas, **volver a compilar (model.compile(...)) el modelo**.
2. **Usar un Learning Rate MUY BAJO:**
   Para evitar "dañar" o "corromper" esos pesos óptimos muy rápidamente, haciendo que el modelo empeore en lugar de mejorar.
3. **Re-entrenar el Modelo Completo (o casi completo):**
   Ahora, el model.fit() entrena tanto las capas personalizadas como las capas de VGG16 que se acaba de descongelar. El objetivo es que estas capas de VGG16 se **adapten mejor a las imágenes de radiografías**, aprendiendo características que son más discriminativas para la neumonía y las imágenes normales, lo que debería llevar a un mejor rendimiento general, especialmente en métricas como el **Recall**.

Evaluación del Modelo Final con Fine-Tuning

Tras el **Fine-Tuning**, el modelo de Transfer Learning (VGG16) mostró un rendimiento **excelente y equilibrado** en el conjunto de prueba de 468 imágenes.

Métricas Clave (calculadas por Keras):

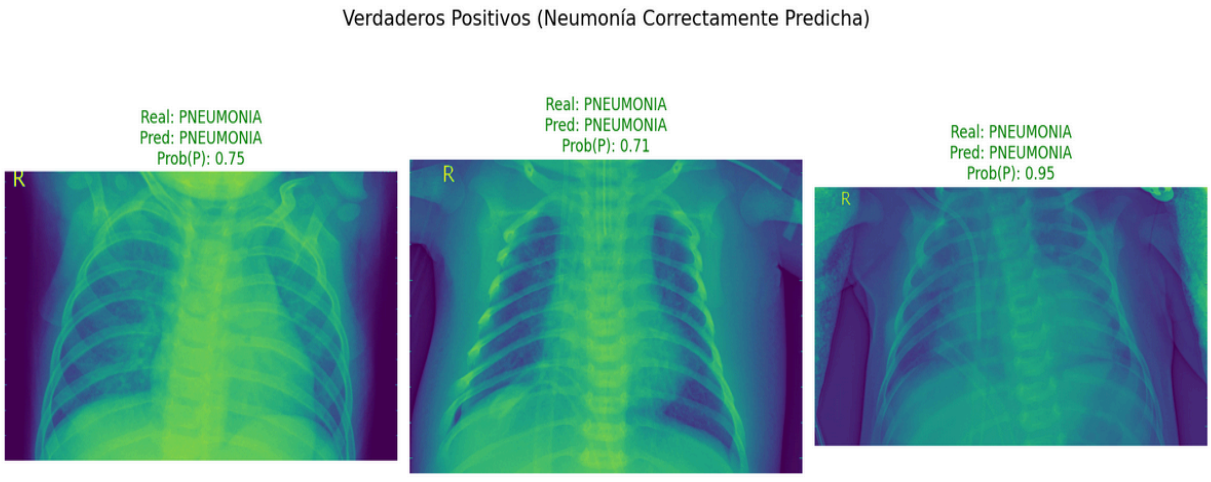
- **Accuracy (Precisión General):** 91.03%
- **Precision (Predicciones Positivas correctas):** 91.38%
- **Recall (Detección de Neumonía real):** 90.60%
- **AUC (Capacidad de Discriminación):** 0.9608

Análisis de Errores Específicos:

- De las 468 imágenes de prueba:
  - **131 casos de Neumonía** fueron correctamente identificados.
  - **228 casos Normales** fueron correctamente identificados.
  - Solo **6 casos Normales** fueron erróneamente clasificados como Neumonía (Falsos Positivos).
  - **103 casos de Neumonía** no fueron detectados por el modelo (Falsos Negativos).

Conclusión: Las métricas de Keras (Accuracy, Precision, Recall por encima del 90% y un AUC alto) demuestran que el modelo de Transfer Learning con Fine-Tuning es altamente efectivo y fiable para la detección de neumonía.

6. Conclusiones



El proceso iterativo de modelado ha confirmado que el sobreajuste fue el principal obstáculo en el desarrollo del modelo. Las reiteradas modificaciones en la arquitectura de la CNN desde cero (filtros, learning_rate, regularización, pooling) no lograron superar este problema fundamental de generalización.

Finalmente, la adopción de la estrategia de **Transfer Learning con VGG16** resultó ser la solución definitiva. El modelo resultante es robusto y ofrece un rendimiento prometedor, especialmente en términos de neumonía, lo que reduce las alarmas falsas.

Consideraciones Clínicas: Para una aplicación práctica, el modelo necesitaría una validación rigurosa por parte de expertos médicos y una calibración del umbral de decisión para adaptarse a necesidades clínicas específicas (ej., si se prefiere un mayor Recall a expensas de una ligera disminución en la Precision).