Proyecto MLOps: Clasificación de Neumonía en Radiografías con TensorFlow, MLflow y Docker

Grupo 5 - ML-Ops y Puesta en Producción - MIAV1E1

Esperanza Miriam Aquino Mamani
Estudiante
UAGRM
La Paz, Bolivia

Carlos Jimy Cespedes Mendez
Estudiante
UAGRM
Santa Cruz, Bolivia

Guido Francisco Ticona Barros
Estudiante
UAGRM
Santa Cruz, Bolivia

Roger Diego Flores Condori
Estudiante
UAGRM
La Paz, Bolivia

***Abstract*—Este proyecto implementa un pipeline de MLOps para el despliegue de un modelo de clasificación de neumonía en radiografías de tórax utilizando redes neuronales convolucionales (CNN). Se utilizó el dataset NIH Chest X-ray, para entrenar y evaluar el modelo. El enfoque principal consistió en aplicar buenas prácticas de MLOps —como control de versiones de datos y código (DVC y Git), registro de experimentos (MLflow), pruebas automatizadas (Unittest), contenerización (Docker) y despliegue de una API REST (Flask)— para garantizar trazabilidad, reproducibilidad y escalabilidad del flujo de trabajo.**

El objetivo central fue validar la eficacia de la metodología MLOps en lugar de optimizar exhaustivamente el rendimiento del modelo. Los resultados destacan la viabilidad de implementar sistemas de IA en entornos clínicos con infraestructura confiable y mantenible, sentando las bases para futuras iteraciones y mejora continua.

Palabras clave: MLOps, deep learning, clasificación de imágenes, neumonía, radiografías de tórax, TensorFlow, Docker, MLflow.

I. Introducción

El área de investigación seleccionada para este proyecto es la salud, con un enfoque en el apoyo al diagnóstico médico mediante inteligencia artificial. En particular, se abordó el problema de la detección de neumonía a partir de imágenes de rayos X de tórax, una enfermedad respiratoria común y potencialmente grave, cuya identificación temprana es crucial para un tratamiento oportuno.

El objetivo principal fue desarrollar un modelo predictivo que pudiera asistir a profesionales de la salud en la interpretación de radiografías, reduciendo la dependencia de la evaluación visual humana y minimizando errores de diagnóstico.

Se utilizó un conjunto de datos públicos de radiografías de tórax para entrenar una red neuronal convolucional. Los resultados mostraron un alto desempeño del modelo, con una precisión del 93% y una capacidad para detectar el 99% de los casos positivos, lo que demuestra su utilidad potencial como herramienta de apoyo clínico.

Este proyecto no solo buscó obtener un modelo preciso, sino también implementar las mejores prácticas de MLOps para garantizar su despliegue eficiente, reproducible y escalable en un entorno productivo..

II. Datos

A. Características y metadatos del dataset:

Se utilizó el dataset NIH Chest X-ray [1], que contiene 112,120 imágenes de rayos X de tórax en formato PNG, con una resolución de 1024 x1024 píxeles. Cada imagen incluye metadatos como:

1) Image Index: Identificador de la imagen.
2) Finding Labels: Etiquetas de patologías (pueden ser múltiples por imagen).
3) Patient ID: Identificador único del paciente.
4) Patient Age y Gender: Edad y género del paciente.
5) View Position: Posición de la radiografía (frontal/lateral).

Las imágenes están anotadas con 14 patologías torácicas, entre ellas: Atelectasia, Cardiomegalia, Derrame, Infiltración, Neumonía, Neumotórax, entre otras.

B. EDA y equilibrio de anotaciones:

El análisis exploratorio mostró que las clases no están equilibradas. La neumonía está presente en aproximadamente 1,300 imágenes, mientras que otras patologías como Infiltración aparecen en más de 9,500. Esto implica un desequilibrio significativo, que puede afectar el entrenamiento del modelo.

Dado que el proyecto en primera instancia tiene objetivos académicos se tomó una muestra mínima de los datos con el fin de configurar el entorno MlOps.

C. Preprocesamiento y tratamiento de datos:

Para mitigar el desbalance, se aplicaron las siguientes técnicas:

1) Filtrado: Se seleccionaron solo las imágenes con etiquetas de "Neumonía" y "No findings" (sanas).
2) Balanceo: Se utilizó aumento de datos (data augmentation) para la clase minoritaria (neumonía), con transformaciones como rotaciones, volteos y ajustes de brillo.

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

3) Reducción de resolución: Las imágenes se redimensionaron a 500x500 píxeles para optimizar el entrenamiento.
4) Normalización: Los valores de píxeles se escalaron al rango [0, 1].

Estos pasos fueron esenciales para mejorar el rendimiento del modelo y evitar sobreajuste hacia las clases mayoritarias..

III. Métodos

A. *Enfoque de solución*

Para abordar el problema de clasificación de neumonía en radiografías de tórax, se implementó un modelo de aprendizaje profundo basado en una red neuronal convolucional (CNN) utilizando transfer learning. La arquitectura seleccionada fue DenseNet121 preentrenada con ImageNet, a la cual se añadieron capas fully connected para adaptarla a la tarea específica. Este enfoque permite aprovechar características genéricas aprendidas de un conjunto de datos grande (ImageNet) y fine-tunearlas para el dominio médico, lo que es especialmente útil given el tamaño limitado de datasets médicos anotados.

B. *Justificación del enfoque*

La elección de DenseNet121 se basa en su eficiencia computacional y capacidad para capturar características multinivel en imágenes, crítico para patrones sutiles en radiografías. El uso de transfer learning reduce el riesgo de overfitting y acelera el entrenamiento. La función de pérdida binary cross-entropy es adecuada para la naturaleza multiclase del problema (varias patologías posibles por imagen), y el optimizador Adam asegura una convergencia estable.

C. *Pipeline de MLOps*

El entrenamiento se integró en un pipeline de MLOps para garantizar reproducibilidad y escalabilidad:

MLflow registra hiperparámetros, métricas por época (pérdida, precisión) y el modelo final.

El entrenamiento se ejecuta en 4 épocas con un batch size de 4, balanceando eficiencia y estabilidad.

D. *Enfoques alternativos considerados para futuras iteraciones*

Modelos arquitectónicamente más complejos (Ej: EfficientNetV2, Vision Transformers) para capturar dependencias de largo alcance en imágenes.

Técnicas de aumento de datos más agresivas (mixup, cutMix) para mejorar la generalización.

Learning rate scheduling o fine-tuning progresivo para afinar capas convolucionales preentrenadas.

Métodos de explicabilidad (Grad-CAM) para generar mapas de calor que justifiquen las predicciones clínicas..

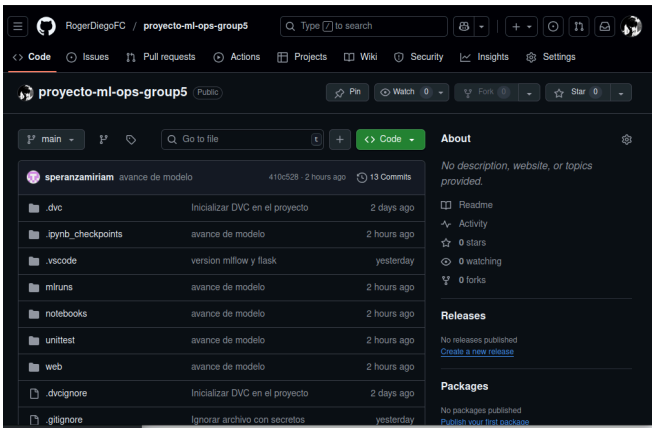
IV. Despliegue

A continuación se detalla la estrategia de despliegue del modelo de clasificación de neumonía en rayos X, implementando las mejores prácticas de MLOps:

A. *Control de Versiones con Git y DVC*

Se utilizó Git para el versionado de todo el código fuente, permitiendo la colaboración del equipo y el mantenimiento de un historial de cambios. El repositorio incluye los scripts de preprocesamiento, entrenamiento, evaluación y despliegue, así como los archivos de configuración.

Para el versionado de datos y modelos se implementó DVC (Data Version Control), que permite rastrear el conjunto de datos y los artefactos generados sin almacenarlos directamente en Git. Los datos originales y preprocesados se almacenaron en un repositorio remoto (Amazon S3), mientras que los archivos .dvc permiten reproducir exactamente cada experimento con los datos correspondientes.

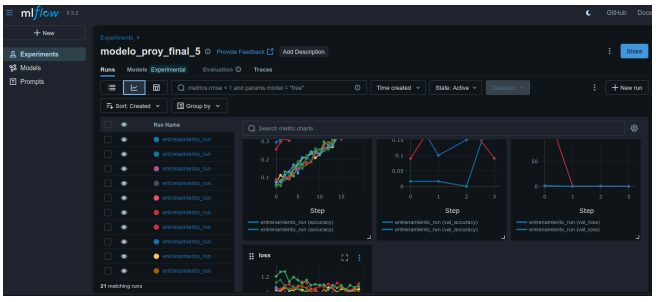


B. *Seguimiento* de Experimentos con MLflow

Se configuró MLflow para el registro y seguimiento sistemático de todos los experimentos. Para cada ejecución se registraron:

- Parámetros: Hiperparámetros del modelo (número de épocas, tasa de aprendizaje, tamaño del batch)
- Métricas: Precisión, recall, pérdida en entrenamiento y validación, F1-score
- Artefactos: Modelos entrenados, gráficas de curvas de aprendizaje.

Cada experimento se registró permitiendo la comparación entre diferentes ejecuciones y la selección del mejor modelo para producción.



C. *Pruebas Unitarias con Unittest*

Se implementaron pruebas unitarias exhaustivas utilizando el framework Unittest de Python para garantizar la robustez y confiabilidad de la aplicación. Las pruebas cubren diferentes aspectos críticos del sistema:

a) Prueba de Humo (Smoke Test):

Verifica que el servicio esté activo y responda correctamente en el endpoint raíz.

Confirma que la API devuelve un código de estado HTTP 200.

b) Prueba de Caso Válido (Single Valid Prediction):

Evalúa el flujo completo de predicción enviando una imagen médica válida.

Valida que la respuesta tenga código 200 y contenga el diagnóstico esperado en el formato correcto.

c) Pruebas de Casos Extremos (Edge Cases):

Verifica el manejo adecuado de entradas incorrectas, como:

- Archivos vacíos o corruptos
- Formatos de imagen no válidos
- Confirma que la API rechaza estas entradas adecuadamente con códigos de error apropiados.

d) Prueba de Consistencia (Pattern Consistency):

Evalúa la consistencia del modelo ante múltiples solicitudes idénticas.

Verifica que imágenes idénticas produzcan exactamente el mismo resultado de diagnóstico

D. *Empaquetamiento* con Docker

Se creó un Dockerfile que define el entorno de ejecución completo:

a) Imagen base oficial de Python 3.9

b) Instalación de dependencias especificadas en requirements.txt

c) Copia del código fuente, modelo entrenado y configuración

d) Exposición del puerto 8085 para la API

e) Definición del comando de inicio para la aplicación

f) La imagen Docker resultante se almacena en Docker Hub, facilitando su distribución y despliegue en cualquier entorno que soporte contenedores.

```
FROM python:3.9-slim

EXPOSE 5002

WORKDIR /app
COPY . /app

# Install pip requirements
COPY requirements.txt .
RUN apt-get update && apt-get install ffmpeg libsm6 libxext6  -y
RUN pip install -r requirements.txt


# During debugging, this entry point will be overridden. For more i
CMD ["gunicorn", "--bind", "0.0.0.0:5002", "app:app"]
```

E. Formulario *con* Flask

Se desarrolló un formulario web y endpoint utilizando Flask con los siguientes endpoints:

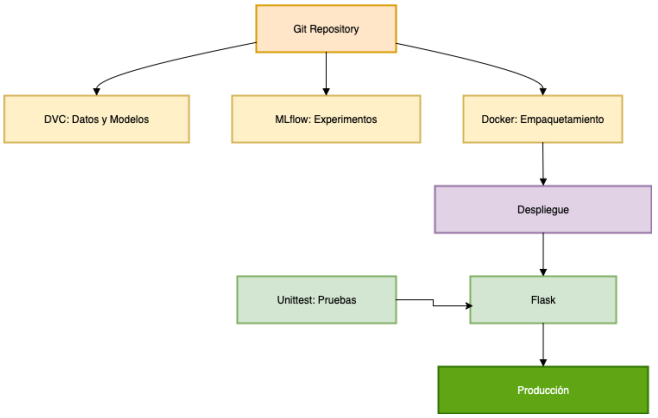
GET / Endpoint de salud que verifica que el servicio esté ejecutándose correctamente

POST /predict: Endpoint de predicción que recibe una imagen en formato JPEG o PNG, la preprocesa, ejecuta la inferencia del modelo y devuelve la clasificación (neumonía o sano) junto con las probabilidades asociadas

El formulario incluye el cargado de la imagen y la visualización del resultado.

F. *Pipeline de MLOps Integrado*

La estrategia de despliegue sigue un flujo que combina todas las herramientas anteriores:



CONCLUSIONES

El desarrollo de este proyecto ha permitido implementar y validar un pipeline completo de MLOps para el despliegue de modelos de deep learning en el ámbito de diagnóstico médico.

A. *Los resultados clave demuestran que:*

Implementación exitosa de prácticas MLOps: Se ha establecido un flujo de trabajo robusto que integra control de versiones (Git/DVC), experimentación y gestión de modelos (MLflow), pruebas automatizadas (Unittest), contenerización (Docker) y despliegue de servicios (Flask). Este ecosistema garantiza orden, escalabilidad y colaboración en el ciclo de vida del modelo.

Reproducibilidad y trazabilidad: El uso de DVC y MLflow asegura que todos los experimentos sean completamente reproducibles y que se mantenga un historial detallado de cada iteración, facilitando auditorías y comparaciones entre versiones del modelo.

Infraestructura escalable: La arquitectura basada en contenedores permite el despliegue consistente en diferentes entornos (local, nube o híbrido), lo que facilita la transición fluida desde desarrollo hasta producción sin dependencia del entorno.

Validación del enfoque: El principal objetivo fue demostrar la viabilidad de las prácticas MLOps más que optimizar al máximo el rendimiento del modelo. Este resultado confirma que la estrategia es aplicable a problemas reales en diagnóstico médico asistido por IA.

B. Aspectos mejorables identificados:

Optimización del modelo: El modelo actual constituye una versión base. Requeriría un ajuste más exhaustivo de hiperparámetros, exploración de arquitecturas alternativas y técnicas de regularización para alcanzar un mejor rendimiento en producción.

Sistema de monitoreo en producción: Resulta necesario implementar herramientas específicas para monitorizar el drift de datos, detectar anomalías y evaluar continuamente el rendimiento del modelo en tiempo real.

Pipeline CI/CD completo: La integración con herramientas de integración y despliegue continuo (por ejemplo, GitHub Actions, GitLab CI o Jenkins) permitiría automatizar la validación, pruebas, construcción de imágenes y despliegue, mejorando así la eficiencia del proceso y reduciendo riesgos de errores humanos.

Este proyecto sirve como demostración práctica de cómo las prácticas MLOps pueden aplicarse sistemáticamente, sentando las bases para implementaciones más complejas en entornos de misión crítica. La experiencia obtenida resalta que la infraestructura operacional resulta tan crucial como el desarrollo del modelo mismo, y que la integración temprana de buenas prácticas de MLOps acelera la transición desde prototipos académicos hacia soluciones de valor real en el ámbito médico.

Referencias

[1] National Institutes of Health Chest X-Ray Dataset https://www.kaggle.com/datasets/nih-chest-xrays/data/data?select=Data_Entry_2017.csv

[2] Lesia Mochurad Classification of X-Ray Images of the Chest Using Convolutional Neural Networks., https://ceur-ws.org/Vol-3038/paper17.pdf.