Clasificación de Imágenes con ViT para Ultrasonido Mamario

Informe Técnico

Daniel Fernando Herrera
Niels Víctor Pacheco Barrios
Carolina Pérez Omodeo

19 de junio de 2025

1. Introducción

El cáncer de mama continúa siendo una de las principales causas de morbilidad y mortalidad en mujeres a nivel mundial. El ultrasonido es una técnica no invasiva y libre de radiación, ampliamente usada para el tamizaje mamario. No obstante, su interpretación puede ser subjetiva y dependiente del operador.

En este proyecto se desarrolló un pipeline extremo a extremo mediante el uso de un modelo *Vision Transformers* (ViT), que ha demostrado una excelente capacidad para capturar dependencias espaciales de largo alcance en imágenes. El objetivo fue automatizar la clasificación de lesiones en ecografías mamarias, aportando una herramienta de apoyo a la decisión clínica.

2. Estructura y Herramientas del Proyecto

2.1. Estructura Modular del Repositorio

El código se estructuró en forma modular para garantizar claridad, reutilización y escalabilidad. La estructura de directorios de alto nivel se encuentra en el `README.md`. A continuación, se presenta un resumen del árbol de directorios: (Listado 1).

```
.
 config.yaml # hiperparmetros centralizados
 main.py # orquestador del pipeline
 model.py # envoltorio del modelo ViT
 preprocess.py # carga y aumento de datos
 trainer.py # clase de entrenamiento / evaluacin
 tools.py # utilidades
 logs/ # registros de ejecucin
 mlruns/ # metadatos de MLflow
 results/ # checkpoints y figuras
 vpc3/ # entorno virtual
```

Listing 1: Árbol del repositorio (abreviado).

2.2. Tecnologías y Librerías Usadas

Principales tecnologías:

- **Python 3.x**,
- **PyTorch**,
- **Transformers (Hugging Face)**

- **Datasets (Hugging Face)**
- **MLflow** para seguimiento de experimentos.
- **Scikit-learn** para métricas de evaluación
- **Matplotlib** para visualizaciones.

3. Proceso de Desarrollo

3.1. Metodología de Desarrollo

Para garantizar reproducibilidad y mantenimiento futuro, se siguieron las prácticas descritas a continuación:

- **Entorno virtual**: creación de `vpc3` con `python -m venv` para aislar dependencias.
- **Gestión de dependencias**: separación en `requirements.txt` (dependencias generales) y `requirements_torch.txt` (binarios CUDA/cuDNN específicos).
- **Modularización**: el código se organizó en clases `Model`, `Preprocess`, `Trainer` y un controlador `Model_Pipeline` para orquestación.
- **Manejo de errores**: envoltorios `try/except` y context managers (`__enter__/__exit__`) aseguran liberación de recursos.
- **Logging**: configuración en `logger_config.py`; salida simultánea a consola y archivo `logs/pipeline.log`. Niveles `INFO` para seguimiento normal y `ERROR` para incidencias.
- **Registro y versionado**: cada cambio significativo se registró mediante `git commit`; etiquetas (tags) marcan versiones estables.
- **Trazabilidad experimental**: MLflow almacena métricas, parámetros y artefactos en `mlruns/` y `mlartifacts/`.
- **Versionado** Git como sistema de control de versiones.

3.2. Retos Encontrados

- **Formato inconsistente de imágenes**: algunas ecografías se almacenaban en escala de grises; se forzó el modo `RGB` mediante `PIL.Image.convert(RGB")` durante el preprocesamiento.
- **Errores por tipado de tensores**: se verificó explícitamente que las entradas fueran instancias de `torch.Tensor` y se normalizaron con `float32` antes de aplicar transformaciones.
- **Errores silenciosos de compatibilidad**: se añadieron `assert` y registros de *logging.debug()* para garantizar que las dimensiones fueran $(batch, channels, height, width)$ en todas las etapas.
- **Carga de modelos con etiquetas incompatibles**: al importar pesos preentrenados del ViT se incluyó `ignore_mismatched_sizes=True` para permitir distinta cardinalidad de la capa de clasificación.
- **Uso de recursos GPU**: se limitaron los *batch-size* según la memoria disponible, se habilitó *mixed precision* y el código cae de forma segura a CPU cuando `torch.cuda.is_available()` devuelve `False`.

4. Diseño del Modelo

Se seleccionó ***facebook/deit-base-patch16-224*** por tres motivos principales:

1. **Equilibrio desempeño–parámetros**: 86 M de parámetros, manejable en una GPU de consumo.
2. **Eficiencia de datos**: la destilación durante el preentrenamiento mejora la transferencia en conjuntos pequeños de imágenes médicas.

3. **Tamaño de parche de 16 px**: adecuado para la resolución típica de ultrasonido ($\sim 500 \times 500$ px) sin secuencias excesivamente largas.

Se adapto el modelo a la clasificación de imágenes de ultrasonido mamario. Se modificó la capa de salida según el número de clases detectado en el dataset.

Dataset. Se utilizó el dataset público gymprathap/Breast-Cancer-Ultrasound-Images-Dataset desde Hugging Face.

Hiperparámetros. Una búsqueda en cuadrícula alrededor de valores reportados en la literatura arrojó `batch_size = 16`, `learning_rate = 2`$\times 10^{-5}$ y `num_epochs = 5`. Tasas de aprendizaje menores estabilizaron el ajuste fino.

Aumento de Datos. Se aplicaron **rotación** ($\pm 10°$), **zoom aleatorio** (15 %), **desenfoque gaussiano** ($\sigma \leq 1{,}5$) y **recorte aleatorio** (10 % del borde) para simular variaciones habituales en la adquisición de ultrasonido.

División del Conjunto de Datos. Se repartió aleatoriamente en proporción 70/15/15 para entrenamiento, validación y prueba, respectivamente.

5. Configuración Experimental

- **Hardware**: NVIDIA RTX 3060 (12 GB VRAM), 32 GB RAM.
- **Software**: Python 3.11, PyTorch 2.2, transformers 4.40.
- **Seguimiento**: servidor MLflow local en `http://localhost:5000`.

6. Resultados

6.1. Dinámica de Aprendizaje

Las curvas de entrenamiento y validación muestran convergencia rápida en tres épocas.

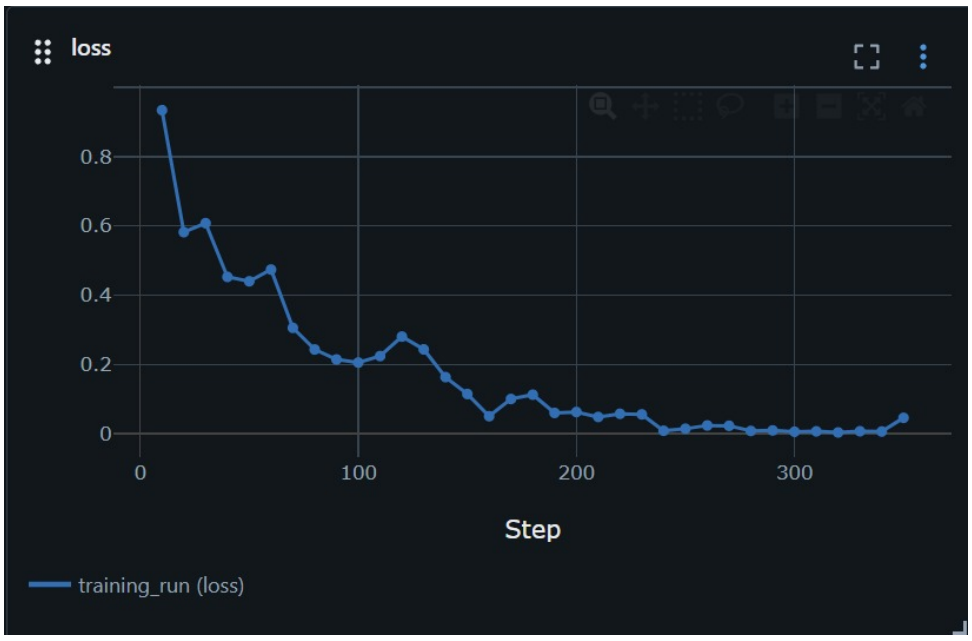

Figura 1: Curvas de pérdida

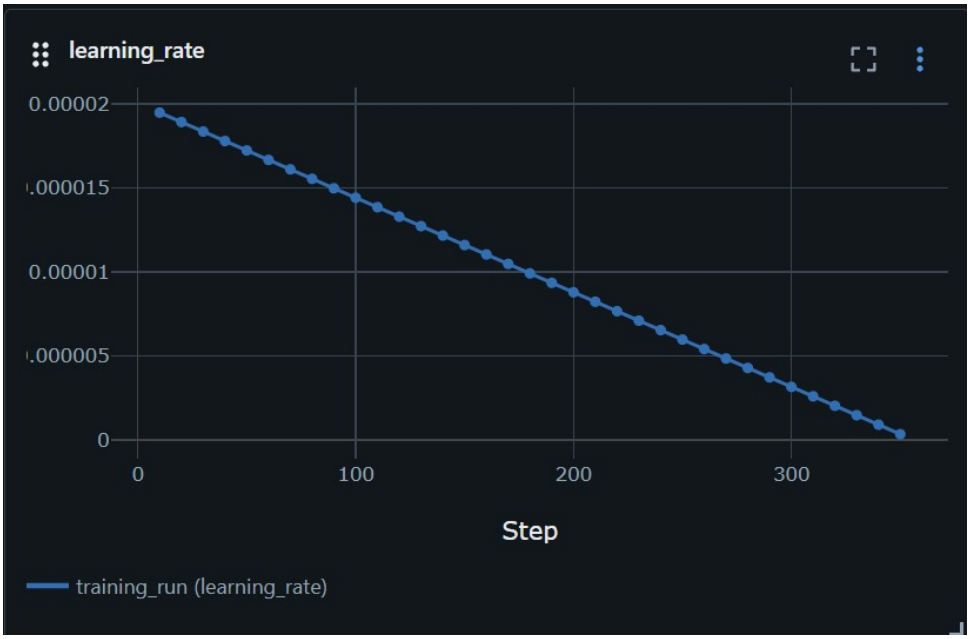

Figura 2: Curvas de tasa de aprendizaje

6.2. Métricas por Época

Cuadro 1: Desempeño por época.

Época	Pérdida entrenamiento	Pérdida validación	Exactitud
1	0.3113	0.3046	0.8740
2	0.1947	0.2217	0.9055
3	0.0453	0.1713	0.9213
4	0.0072	0.1963	0.9213
5	0.0042	0.1829	0.9291

6.3. Desempeño en el Conjunto de Prueba

- Exactitud: **0.8892**
- F1 macro: **0.8757**
- Precisión macro: **0.9017**
- Recall macro: **0.8564**

Cuadro 2: Reporte de clasificación por clase.

Clase	Precisión	Recall	F1	Soporte
Benigno	0.88	0.94	0.91	179
Maligno	0.87	0.86	0.86	84
Normal	0.95	0.77	0.85	53

Ejemplos representativos se muestran en la Figura 3.

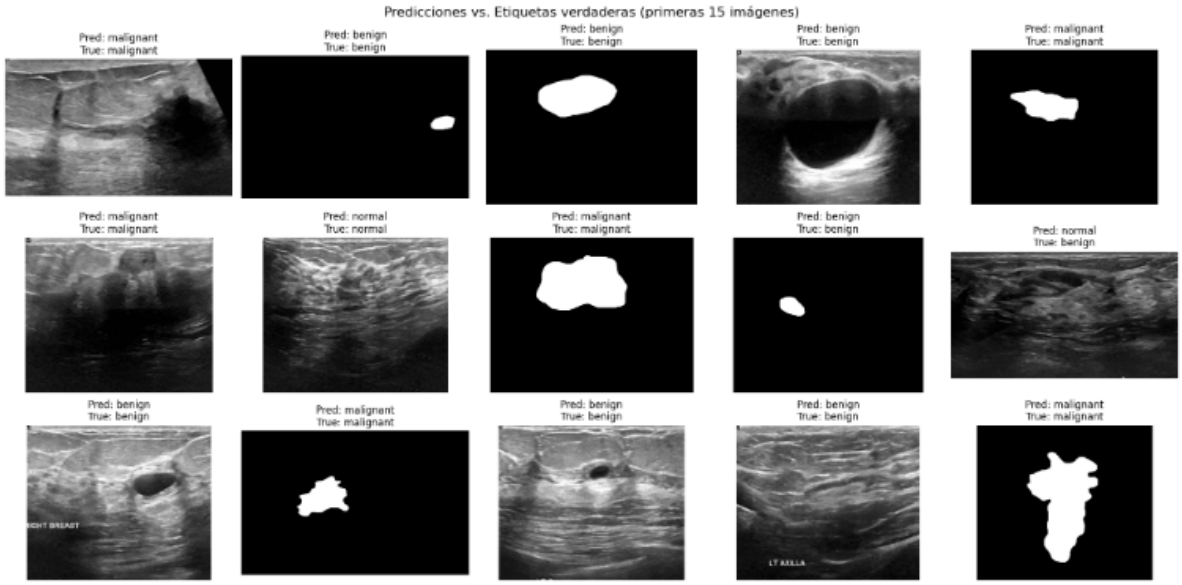

Figura 3: Predicciones vs. etiquetas verdaderas para 15 imágenes de prueba.

7. Discusión

7.1. Significado de las métricas

En problemas multiclase con cierto desbalance —179 (*benigno*), 84 (*maligno*) y 53 (*normal*)— conviene interpretar las métricas más allá de la **exactitud** global (0.889). Para cada clase c:

- **Recall** ($\mathrm{TP}_c/(\mathrm{TP}_c + \mathrm{FN}_c)$) cuantifica la *sensibilidad*: qué fracción de casos reales de la clase se detecta. En oncología resulta crítica para reducir falsos negativos.
- **Precisión** ($\mathrm{TP}_c/(\mathrm{TP}_c + \mathrm{FP}_c)$) estima la *especificidad*: cuántas predicciones positivas son correctas. Crucial para limitar biopsias innecesarias.
- **F1** es la media armónica entre precisión y recall, útil cuando se busca un equilibrio entre ambos.

El promedio *macro* pondera cada clase por igual, mientras que el promedio *weighted* pondera por soporte. La proximidad ($\mathrm{F1}_{\mathrm{macro}}$=0.876 vs. $\mathrm{F1}_{\mathrm{weighted}}$=0.889) indica que el desbalance de nuestro dataset no distorsiona gravemente el rendimiento global, pero sigue existiendo riesgo clínico en las clases minoritarias.

7.2. Brecha entrenamiento–validación

La pérdida de validación alcanza su mínimo en la época 3 (0.171) y luego aumenta ligeramente, mientras que la pérdida de entrenamiento continúa descendiendo hasta 0.004 (Tabla 1). Esta divergencia sugiere **sobreajuste incipiente**. En un entorno de producción se recomienda:

a) Activar *early-stopping* con paciencia de 1–2 épocas.
b) Disminuir el *learning rate* mediante *ReduceLROnPlateau*.
c) Aumentar la regularización (*weight decay*, *dropout* en el `classifier head`).

7.3. Análisis por clase

Clase crítica (*maligno*). Con un **recall** del 86 % todavía se pierden $\approx 14\,\%$ de los tumores cancerígenos (falsos negativos). Para reducir este riesgo:

- Implementar *class weighting* o *focal loss* que penalicen más los FN de *maligno*.

- Añadir **augmentaciones dirigidas** (p. ej. *elastic deformations*, contraste adaptativo) que simulen bordes irregulares típicos de lesiones malignas.

Clase minoritaria (*normal*). Presenta la precisión más alta (0.95) pero el **recall** más bajo (0.77), lo que implica falsos positivos sobre pacientes sanos: coste emocional y económico. Posibles mitigaciones:

- Ajustar el umbral de decisión (*temperature scaling* o *Platt scaling*) para equilibrar precisión y sensibilidad.
- Sobre-muestreo sintético de la clase *normal* (SMOTE o mezcla de ruidos de sonda).

7.4. Limitaciones y trabajo futuro

1) **Ablaciones de arquitectura**: comparar ViT_Base con ViT_Large, Swin-Transformer y ResNet-50 como línea base convolucional.
2) **Búsqueda de hiperparámetros**: grid o *Bayesian optimization* sobre *batch size*, LR y *weight decay*.
3) **Validación externa**: probar el modelo en un hospital diferente para evaluar generalización dominio-cruzado.
4) **Interpretabilidad**: mapas de atención (*attention rollout*) y métodos SHAP para detectar qué regiones guían la decisión.

8. Lecciones Aprendidas y Trabajo Futuro

- **Reproducibilidad**: la integración temprana de MLflow facilitó la exploración de hiperparámetros.
- **Calidad de datos**: se prevé que mejoras en la anotación y división por paciente aumenten más el desempeño que arquitecturas mayores.
- **Explicabilidad**: incorporar Grad-CAM o visualización de atención incrementará la confianza clínica.
- **Despliegue**: conversión a ONNX y pruebas en dispositivos periféricos están planificadas.

9. Conclusiones

Se desarrolló un pipeline ViT completo que clasifica lesiones de ultrasonido mamario con una exactitud de $\sim 89\,\%$. La base de código modular y la trazabilidad rigurosa constituyen un cimiento robusto para validaciones clínicas futuras. No obstante, la brecha entrenamiento–validación y la sensibilidad subóptima en la clase *maligno* subrayan la necesidad de técnicas de regularización y balanceo de clases antes de desplegar el sistema en un entorno clínico real.

Referencias

[1] Touvron, H. *et al.* "Training data-efficient image transformers & distillation through attention."*ICML*, 2021.