Informe Obligatorio:
Machine Learning en Producción

Facultad de Ingeniería - Bernard-Wand Polak

Juan Andrés Rodríguez - 225931

Valeria Eskenazi - 326684

10 de Julio 2025

Resumen Ejecutivo

Este proyecto desarrolla un sistema completo de Machine Learning para la clasificación de imágenes de productos comerciales, específicamente para detectar la presencia de octógonos en productos alimenticios. El sistema integra técnicas de web scraping, procesamiento de imágenes, redes neuronales convolucionales y despliegue en la nube, siguiendo las mejores prácticas para poner sistemas de Machine Learning en producción.

---

1. Introducción y Definición del Problema

1.1 Contexto

El etiquetado nutricional mediante octógonos negros es una imposición regulatoria de varios gobiernos bajo la excusa de informar a los consumidores sobre productos con alto contenido de azúcar, sodio, grasas saturadas y calorías. Este proyecto aborda la necesidad de automatizar la detección de estos octógonos en imágenes de productos.

1.2 Problema Definido

Objetivo: Desarrollar un modelo de computer vision basado en redes neuronales convolucionales para clasificar imágenes de productos según la presencia o ausencia de octógonos nutricionales.

Tipo de problema: Clasificación binaria

- Clase 1: Imagen contiene uno o más octógonos
- Clase 0: Imagen no contiene octógonos

1.3 Justificación

La detección automática de octógonos nutricionales tiene aplicaciones prácticas en:

- Sistemas de inventario automatizado
- Aplicaciones de salud y nutrición
- Análisis de mercado de productos alimenticios
- Herramientas de apoyo para personas con restricciones dietéticas

1.4 Limitaciones y Desafíos del Problema

La principal limitación de un sistema basado exclusivamente en la detección de octógonos nutricionales radica en la incompletitud del etiquetado regulatorio. Existen productos alimenticios que, a pesar de tener altos contenidos de azúcar, sodio, grasas saturadas o calorías, aún no han incorporado el sistema de etiquetado con octógonos debido a:

- Períodos de transición regulatoria: Los fabricantes pueden tener plazos extendidos para implementar el nuevo etiquetado
- Productos importados: Artículos de otros países que no siguen las mismas regulaciones locales. Muchos incluso traen octógonos que pertenecen a otros marcos regulatorios y deben ser tapados.
- Categorías exentas: Algunos productos pueden estar temporalmente exentos de la normativa.
- Incumplimiento normativo: Casos donde los fabricantes no han actualizado sus etiquetas según la regulación vigente, sobre todo por los sobrecostos que implica en la producción de los mismos.

Implicaciones del problema:

- Falsos negativos: El sistema clasificaría como "saludable" un producto que debería tener octógonos pero no los presenta
- Sesgo en la clasificación: Subestimación de productos no saludables en el dataset
- Limitaciones de aplicabilidad: El modelo sería menos efectivo en mercados con baja adopción del etiquetado

Mitigaciones propuestas:

- Combinar la detección de octógonos con análisis de texto de ingredientes
- Implementar sistemas de validación cruzada con bases de datos nutricionales
- Establecer mecanismos de feedback para identificar casos problemáticos
- Considerar la integración con APIs de información nutricional complementaria

2. Construcción del Dataset

2.1 Fuente de Datos

Se construyó un dataset propio mediante web scraping del sitio https://www.disco.com.uy/almacen, extrayendo productos de la categoría "Almacén" junto con sus metadatos (nombre, descripción, precio, imagen).

2.2 Arquitectura del Sistema de Scraping

La arquitectura para el sistema de scraping está organizado de manera modular dentro del directorio src/, basada en la estructura del Práctico 3 de la materia, con las modificaciones necesarias para adaptarse al caso de estudio:

```
src/
├── connectors/
│   ├── openai_client.py        # Conexión con la API de OpenAI
│   └── s3_client.py            # Cliente para interactuar con AWS S3
│
├── scrapers/
│   └── disco.py                # Lógica para scrapear productos de Disco
│
├── scripts/
│   └── empty_s3_bucket.py      # Script auxiliar para vaciar un bucket de S3
│
├── settings/
│   ├── config.yml              # Configuraciones generales del proyecto
│   ├── logger.py               # Configuración de logging
│   └── settings.py             # Lectura de configuraciones
│
├── structs/
│   ├── product.py              # Definición de estructura de productos
│   └── storage_type.py         # Tipos de almacenamiento disponibles
│
├── tags/
│   └── etiquetador.py          # Lógica para el etiquetado automático
│
├── utils/
│   └── io_utils.py             # Utilidades para lectura y escritura de archivos
│
├── main.py                     # Script principal (no utilizado en la ejecución estándar)
└── pipeline.py                 # Script principal de ejecución: scraping, tagging y subida a S3
```

2.3 Proceso de Extracción

Herramienta: Playwright (manejo de contenido dinámico con JavaScript)

Desafío técnico: El sitio web utiliza scroll infinito, requiriendo automatización del desplazamiento:

```
# Scroll para cargar más productos
for _ in range(100):
    page.mouse.wheel(0, 5000)
    page.wait_for_timeout(1000)
```

Con un rango de 100, se logró scrapear 777 imagenes. De requerir un número mayor, se debe aumentar este valor.

Comando de ejecución:

python src/pipeline.py --scrape --upload --bucket 1000-imagenes-scrapper-obligatorio-ml

2.4 Etiquetado Automático (Weak Supervision)

Siguiendo la metodología de "supervisión débil" descrita en "Diseño de sistemas de machine learning" de Chip Huyen, se utilizó la API de OpenAI para el etiquetado automático de imágenes, evitando el proceso manual y costoso de etiquetado.

La instrucción que se le solicito fue: "Dado un envase de producto, responde sólo 'con_octogono' o 'sin_octogono' según si hay octógonos de advertencia nutricional visibles. No expliques nada."

Ventajas:

- Escalabilidad en el etiquetado
- Consistencia en los criterios de clasificación
- Reducción significativa del tiempo de preparación de datos

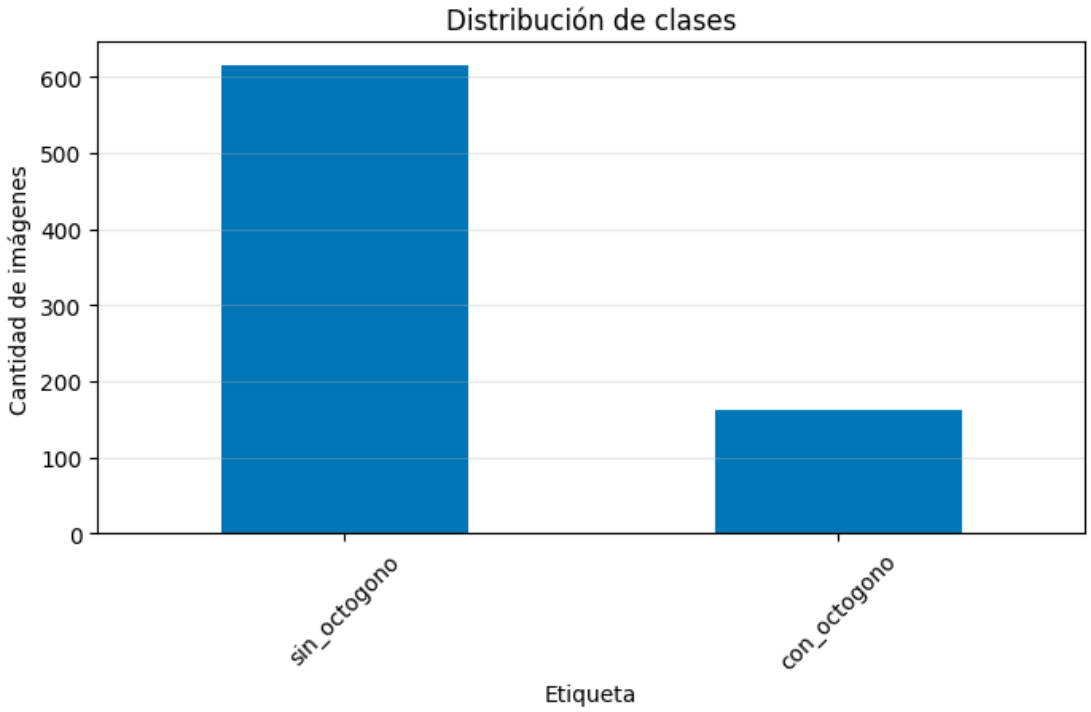
Comando de ejecución:

python src/pipeline.py --tag --upload --bucket 1000-imagenes-scrapper-obligatorio-ml --prefix data/scraped_data/images/

2.5 Análisis Exploratorio de Datos (EDA)

El dataset final construido contiene 777 imágenes de productos alimenticios extraídas del sitio web de Disco Uruguay.

2.5.1 Distribución de Clases



El análisis de la distribución de clases revela un desbalance significativo en el dataset:

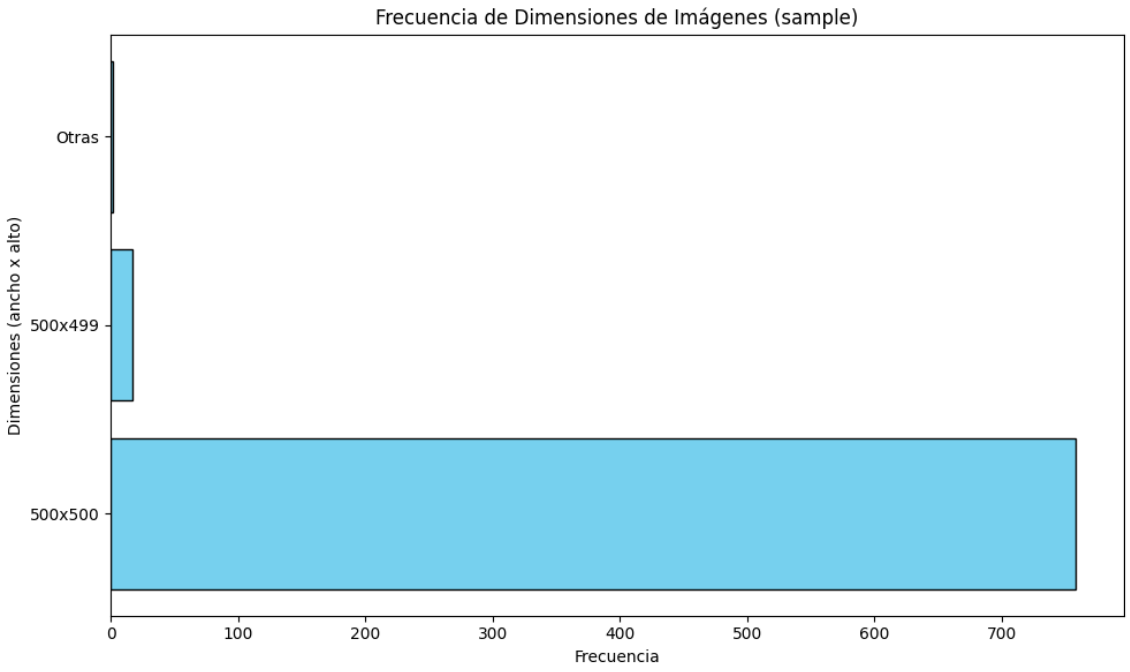
- Sin octógono: ~620 imágenes (79.8%)
- Con octógono: ~157 imágenes (20.2%)

Ratio de desbalance: Aproximadamente 4:1 en favor de productos sin octógonos.

Implicaciones del desbalance:

- Sesgo potencial hacia la clasificación de productos como "sin octógono"
- Necesidad de técnicas de balanceamiento durante el entrenamiento
- Métricas de evaluación que consideren el desbalance (F1-Score)

2.5.2 Análisis de Dimensiones de Imágenes



El estudio de las dimensiones de las imágenes muestra:

- Resolución predominante: 500x500 píxeles (~750 imágenes, 96.5%)
- Resolución secundaria: 500x499 píxeles (~20 imágenes, 2.6%)
- Otras dimensiones: Casos minoritarios (~7 imágenes, 0.9%)

Ventajas de la uniformidad:

- Consistencia en la calidad de imagen
- Procesamiento simplificado (mínimo redimensionamiento)
- Reducción de variabilidad no deseada en el modelo

2.5.3 Características del Dataset

Se observa, que al haberse scrapeado las imágenes de una página de ventas importante, como lo es el supermercado disco, se cuenta con una buena calidad de imágenes que satisface:

- Resolución estándar adecuada para CNN

- Imágenes con fondo blanco consistente
- Productos centrados en el encuadre
- Iluminación uniforme del sitio web

Sin embargo, se identifican los siguientes desafíos:

- Desbalance de clases
- Diferentes tamaños y posiciones de octógonos en productos positivos

Los mismos, serán atendidos en la búsqueda y ajuste de un modelo de clasificación realizados en la notebook clasificacion_imagenes.ipynb.

3. Arquitectura del Sistema

3.1 Ambiente de Desarrollo

Gestión de dependencias:

- requirements.txt para especificación de librerías
- venv para aislamiento del entorno virtual
- Compatibilidad con macOS Sequoia y Ubuntu (GitHub Codespaces)

Containerización:

- Docker para empaquetado de la aplicación
- Imagen base optimizada para ML workloads
- Configuración reproducible entre entornos

3.2 Infraestructura en la Nube (AWS)

Servicios utilizados:

- Amazon ECR: Registro de contenedores Docker con versionado
- Amazon EC2: Instancias de cómputo para entrenamiento y servicio
- Amazon S3: Almacenamiento distribuido de datos y modelos
    - Estructura de buckets organizada por tipo de datos
    - Versionado automático de datasets
    - Acceso programático mediante SDK

Diagrama de arquitectura:

[Scraping] →

[S3 Raw Data] →

[Preprocessing] →

[S3 Processed] →

[Training] →

[Model Registry] →

[API Service] →

[Gradio Service]

3.3 Versionado y Colaboración

Repositorio: https://github.com/valeriaeskenazi/Obligatorio_ML

- Control de versiones con Git
- Repositorio público en GitHub
- Integración continua con GitHub Actions

4. Prevención de Problemas Comunes en ML

4.1 Prevención de Data Leakage

Estrategias implementadas:

- Separación temprana de conjuntos de entrenamiento y prueba
- Fijación de semilla aleatoria para reproducibilidad
- Validación independiente de transformaciones de datos
- Auditoría de características para evitar "future leakage"

Implementación de división de datos:

```
# Fijamos la semilla para que los resultados sean reproducibles
SEED = 23

torch.manual_seed(SEED)
torch.backends.cudnn.deterministic = True
multiprocessing.cpu_count()
```

```
[16]
#Primero: dividir el dataset
train_df, intermedio_df = train_test_split(df, test_size=0.3, stratify=df['label'], random_state=SEED)
val_df, test_df = train_test_split(intermedio_df, test_size=0.5, stratify=intermedio_df['label'], random_state=SEED)
```

4.2 Prevención de Training-Serving Skew

Dado que durante el entrenamiento, se utilizó la siguiente transformación:

```
BS_500_1 = T.Compose([
    T.Resize((500, 500)),
    T.ToTensor(),
])
```

La misma se adapta al pipe linde de la API para la prevención de training-serving skew

5. Desarrollo del Modelo

5.1 Arquitectura de Red Neuronal

Tipo: Red Neuronal Convolucional (CNN) Framework: PyTorch

En la notebook donde se desarrolló el modelo (clasificador_imagenes.ipynb) se proponen 4 modelos, a experimentar con weights and baise:

- Modelo 1 - LeNet - imagenes de 500x500

Adaptación de un modelo LeNet clásico para imágenes de 500×500 píxeles, compuesta por tres capas convolucionales con activaciones tanh, seguidas de capas totalmente conectadas para clasificación.

- Modelo 2 - LeNet - imagenes de 256x256

Variante de LeNet adaptada a imágenes de 256×256 píxeles, con tres capas convolucionales y activaciones tanh, seguida de capas lineales para clasificación binaria.

- Modelo 3 - ResNet18 - AdaptiveAvgPool2d

Versión simplificada de ResNet-18 con bloques residuales y downsampling progresivo, adaptada a entradas de 500×500 píxeles. Incluye cinco bloques convolucionales con conexiones tipo skip y una capa de pooling adaptativo antes de la clasificación final.

- Modelo 4 - ResNet18 - Adaptativo - Regularización

Versión extendida de ResNet-18 con bloques residuales profundos, normalización por lotes y dropout progresivo. Diseñada para imágenes de 500×500 píxeles, incorpora conexiones residuales ajustadas y capas densas finales para clasificación binaria.

5.2 Proceso de Entrenamiento

Para el entrenamiento se utilizó la función de pérdida cross-entropy con ajuste de pesos para compensar el desbalance de clases. Además, se integró la herramienta Weights & Biases (W&B) para el seguimiento, visualización y gestión de los experimentos.

5.2.1 EXPERIMENTO 1 - Sin Data Augmentation

Para la ejecución del experimento se fijaron los siguientes hiperparámetros:

- Tamaño de batch de 32,
- Tasa de aprendizaje (LR) de 0.001
- Optimizador Adam.

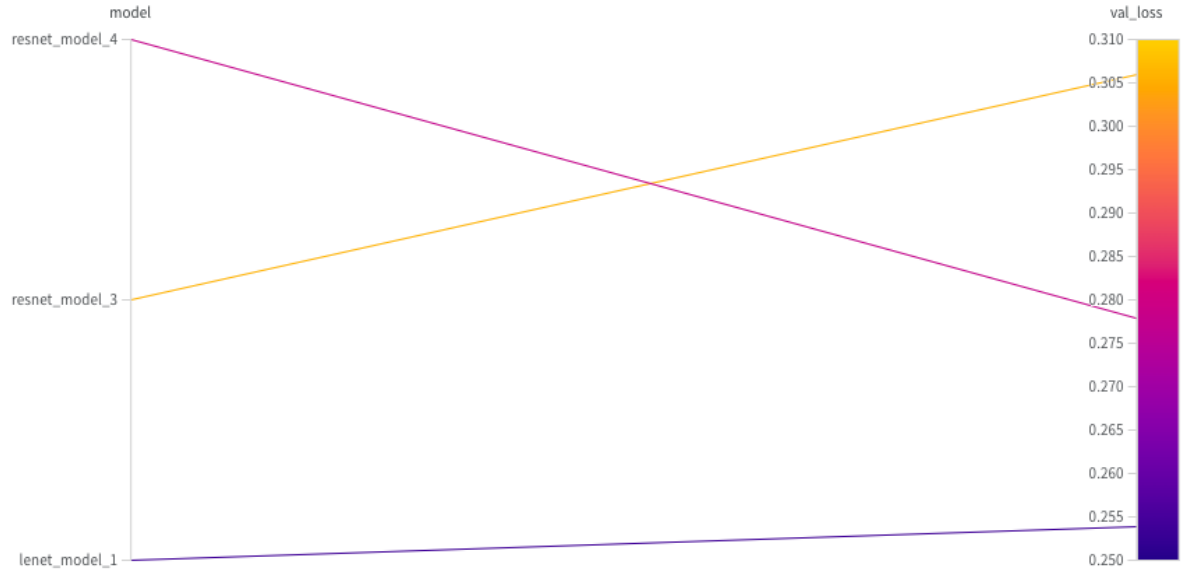
La variable en estudio corresponde a los distintos modelos empleados, manteniendo constante el resto de los parámetros.

El objetivo de este experimento es evaluar el desempeño de cada arquitectura sobre distintos tamaños de imagen, sin aplicar técnicas de data augmentation.

Los resultados obtenidos se presentan a continuación en forma de gráfico, obtenido de la ejecución de W&B y tabla comparativa en donde se imprimió por cada corrida, el reporte con las diferentes métricas.

5.2.1.1 Imagenes de 500x500

Gráfico obtenido de W&B para el experimento 1 con imagenes de 500x500



Conclusiones sobre el gráfico: Se observan diferencias marginales en la función de pérdida para el conjunto de validación. Siendo la mejor, el caso de la red mas simple. Sin embargo, no son suficientemente concluyentes para decir sobre el rendimiento del modelo.

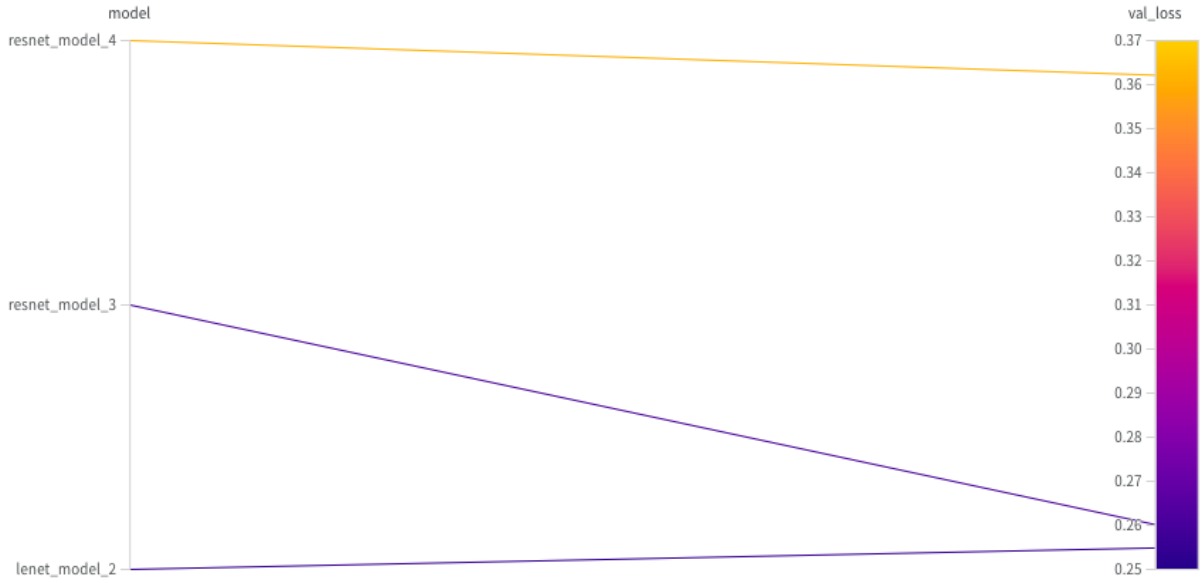
Tabla de metricas:

Modelo	1 - LeNet		3 - ResNet		4 - ResNet	
clases	0	1	0	1	0	1
Precision	0,79	0	0,79	0	0,79	0
Recall	1	0	1	0	1	0
f1-score	0,89	0	0,89	0	0,89	0
support	93	24	93	24	93	24

Conclusiones sobre la tabla: No se observan variación en las métricas, para los distintos modelos en el conjunto de test. Por lo que se concluye que es preciso avanzar con otras técnicas de optimización y mejoras del modelo.

5.2.1.2 Imagenes de 256x256

Gráfico obtenido de W&B para el experimento 1 con imagenes de 256x256



Conclusiones sobre el gráfico: Es casi analogo al anterior, salvanamdo un mejor resultado, en cuento a la funcion de perdida para el conjunto de validacion, en ResNet modelo 3 (modelo simple).

Tabla de metricas

Modelo	1 - LeNet		3 - ResNet		4 - ResNet	
clases	0	1	0	1	0	1
Precision	0,79	0	0,79	0	0,79	0
Recall	1	0	1	0	1	0
f1-score	0,89	0	0,89	0	0,89	0
support	93	24	93	24	93	24

Conclusiones sobre la tabla: Nuevamente, no se observan variación en las métricas, para los distintos modelos en el conjunto de test.

5.2.1.3 Conclusiones del Experimento 1:

Si bien se observaron variaciones en la loss de validación entre modelos, estas no resultaron suficientes como para reflejar diferencias significativas en el desempeño sobre el conjunto de test.

En esta etapa, no es posible extraer conclusiones firmes, aunque se destaca que un modelo simple como LeNet mostró una mejor adaptabilidad inicial, incluso en ausencia de técnicas de optimización o data augmentation.

5.2.2 EXPERIMENTO 2 - Data augmentation

Se mantienen constantes los parámetros definidos en el Experimento 1, pero en este caso se incorpora el uso de data augmentation.

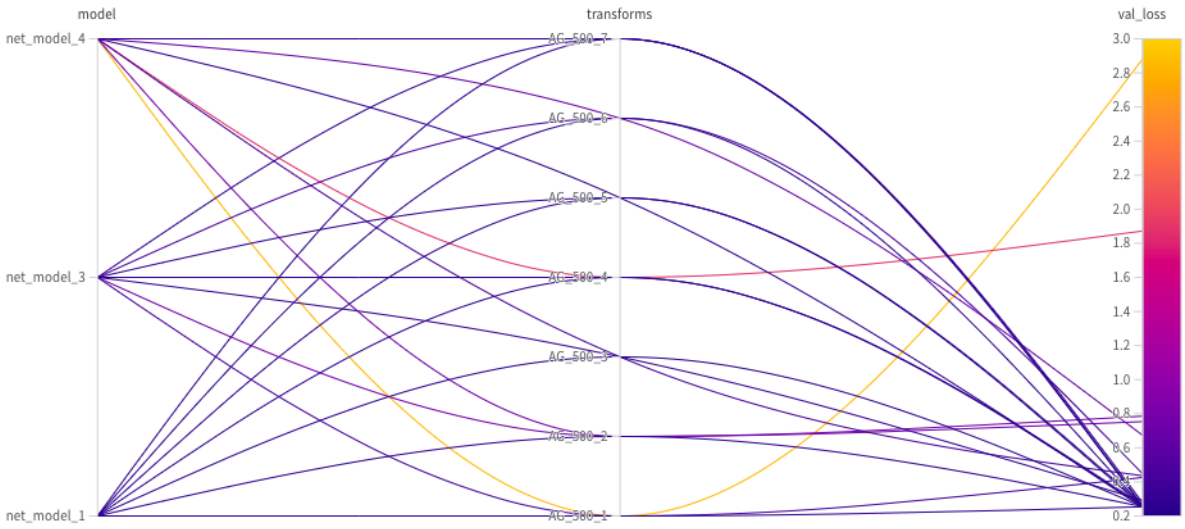
Las técnicas aplicadas para aumentar la variabilidad del conjunto de entrenamiento se resumen en la siguiente tabla:

AG_500_1	T.Resize((500, 500)),	AG_256_1	T.Resize((256, 256)),
	T.RandomHorizontalFlip(p=0.5),		T.RandomHorizontalFlip(p=0.5),
	T.RandomRotation(degrees=15),		T.RandomRotation(degrees=15),
	T.ColorJitter(brightness=0.2, contrast=0.2),		T.ColorJitter(brightness=0.2, contrast=0.2),
AG_500_2	T.Resize((500, 500)),	AG_256_2	T.Resize((256, 256)),
	T.RandomHorizontalFlip(p=0.5),		T.RandomHorizontalFlip(p=0.5),
	T.RandomRotation(degrees=15),		T.RandomRotation(degrees=15),
AG_500_3	T.Resize((500, 500)),	AG_256_3	T.Resize((256, 256)),
	T.RandomRotation(degrees=15),		T.RandomRotation(degrees=15),

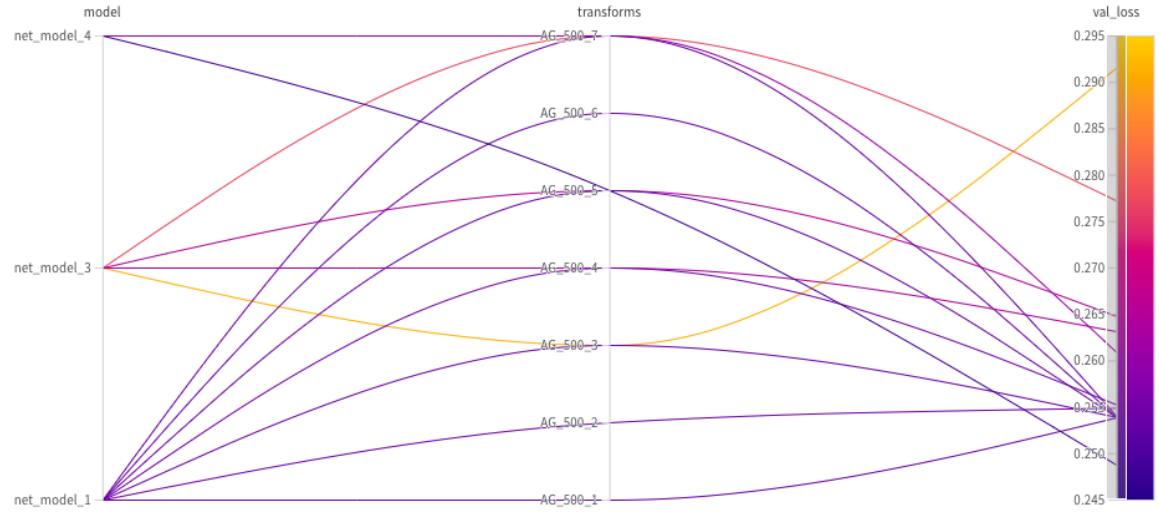
	T.ColorJitter(brightness=0.2, contrast=0.2),		T.ColorJitter(brightness=0.2, contrast=0.2),
AG_500_4	T.Resize((500, 500)),	AG_256_4	T.Resize((256, 256)),
	T.RandomHorizontalFlip(p=0.5),		T.RandomHorizontalFlip(p=0.5),
	T.ColorJitter(brightness=0.2, contrast=0.2),		T.ColorJitter(brightness=0.2, contrast=0.2),
AG_500_5	T.Resize((500, 500)),	AG_256_5	T.Resize((256, 256)),
	T.RandomHorizontalFlip(p=0.5),		T.RandomHorizontalFlip(p=0.5),
AG_500_6	T.Resize((500, 500)),	AG_256_6	T.Resize((256, 256)),
	T.RandomRotation(degrees=15),		T.RandomRotation(degrees=15),
AG_500_7	T.Resize((500, 500)),	AG_256_7	T.Resize((256, 256)),
	T.ColorJitter(brightness=0.2, contrast=0.2),		T.ColorJitter(brightness=0.2, contrast=0.2),

5.2.2.1 Imagenes de 500x500

Gráfico obtenido de W&B para el experimento 2 con imagenes de 500x500



<u>Conclusiones sobre el gráfico:</u> Se observa una gran convergencia de la funcion de perdida para la mayoria de los casos. Se aplica un filtro para poder visualizar mejor;



El mejor resultado lo obtuvo el modelo 4 de la red ResNet, el cual es la red mas compleja, y en particular, para la transformacion AG_500_5.

Tabla de metricas

Transformacion	AG_500_1		AG_500_2		AG_500_3		AG_500_4		AG_500_5		AG_500_6		AG_500_7	
Modelo	1 - LeNet		1 - LeNet		1 - LeNet		1 - LeNet		1 - LeNet		1 - LeNet		1 - LeNet	
clases	0	1	0	1	0	1	2	3	0	1	0	1	0	1
Precision	0,79	0	0,79	0	0,79	0	0,79	0	0,79	0	0,79	0	0,79	0
Recall	1	0	1	0	1	0	1	0	1	0	1	0	1	0
f1-score	0,89	0	0,89	0	0,89	0	0,89	0	0,89	0	0,89	0	0,89	0
support	93	24	93	24	93	24	93	24	93	24	93	24	93	24
Transformacion	AG_500_1		AG_500_2		AG_500_3		AG_500_4		AG_500_5		AG_500_6		AG_500_7	
Modelo	3 - ResNet		3 - ResNet		3 - ResNet		3 - ResNet		3 - ResNet		3 - ResNet		3 - ResNet	
clases	0	1	0	1	0	1	0	1	0	1	0	1	0	1
Precision	0,79	0	0,79	0	0,79	0	0,79	0	0,79	0	0,79	0	0,79	0
Recall	1	0	1	0	1	0	1	0	1	0	1	0	1	0
f1-score	0,89	0	0,89	0	0,89	0	0,89	0	0,89	0	0,89	0	0,89	0
support	93	24	93	24	93	24	93	24	93	24	93	24	93	24
Transformacion	AG_500_1		AG_500_2		AG_500_3		AG_500_4		AG_500_5		AG_500_6		AG_500_7	
Modelo	4 - ResNet		4 - ResNet		4 - ResNet		4 - ResNet		4 - ResNet		4 - ResNet		4 - ResNet	
clases	0	1	0	1	0	1	0	1	0	1	0	1	0	1
Precision	0,79	0	0.80	0,23	0,79	0	1	0,21	0,8	1	0,79	0	0,79	0
Recall	1	0	0.89	0,12	0,98	0	0,02	1	1	0,04	1	0	1	0
f1-score	0,89	0	0,84	0,16	0,88	0	0,04	0,35	0,89	0.08	0,89	0	0,89	0
support	93	24	93	24	93	24	93	24	93	24	93	24	93	24

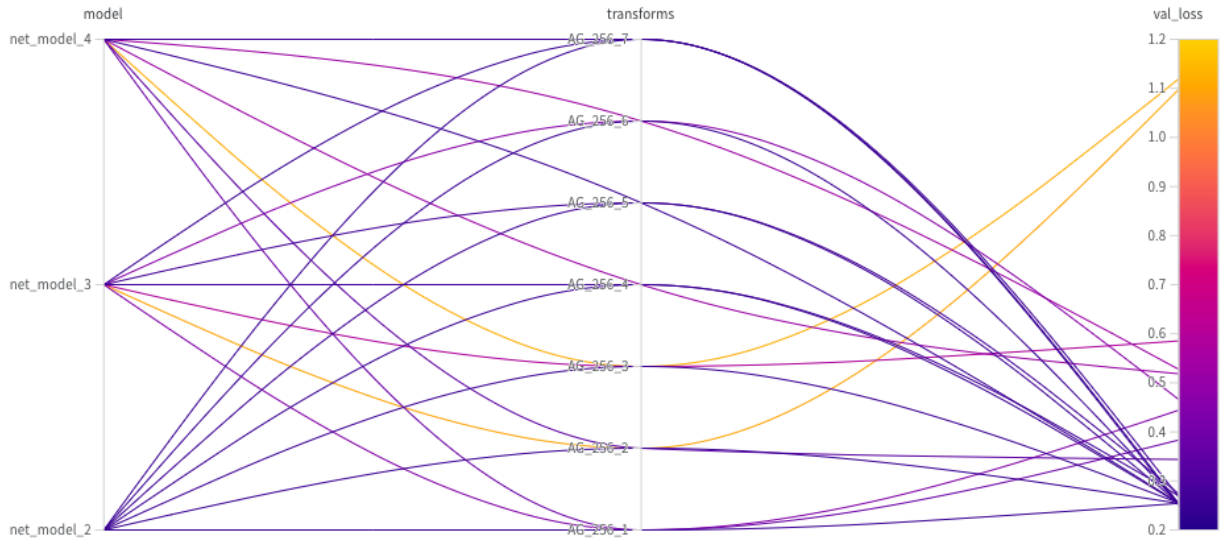
Conclusiones de la Tabla: Al analizar las métricas, se confirma lo observado en el gráfico: únicamente el Modelo 4 presenta variaciones significativas.

Esto sugiere que es el único con capacidad de adaptación, ya que logra captar la variabilidad de los datos incluso en condiciones aparentemente subóptimas.

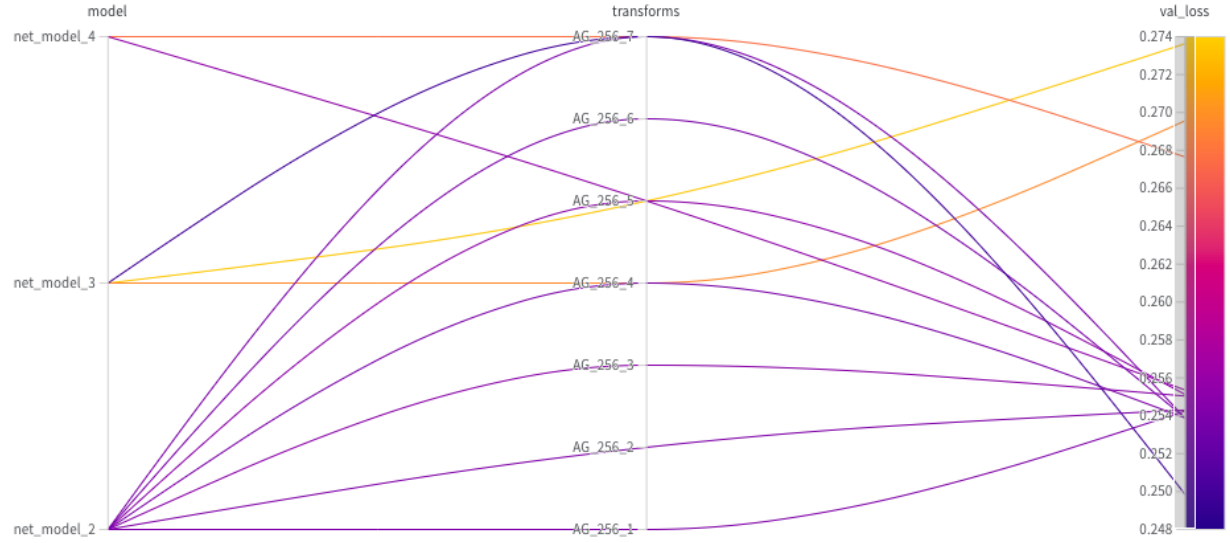
También se concluye, que la transformación AG_500_5, fue la de mejores resultados.

5.2.2.2 Imagenes de 256x256

Gráfico obtenido de W&B para el experimento 2 con imagenes de 256x256



Conclusiones del grafico: Nuevamente, se observa mayor convergencia a valores pequeños de la funcion de perdida, y es preciso, nuevamente, aplicar un filtro para una mejora de la visualizacion:



En este caso, se observa que el mejor valor, lo obtiene el modelo 3, correspondiente a una red ResNet simple, y con la transformacion AG_256_7, como se ilustra en el grafico a acontinuacion:

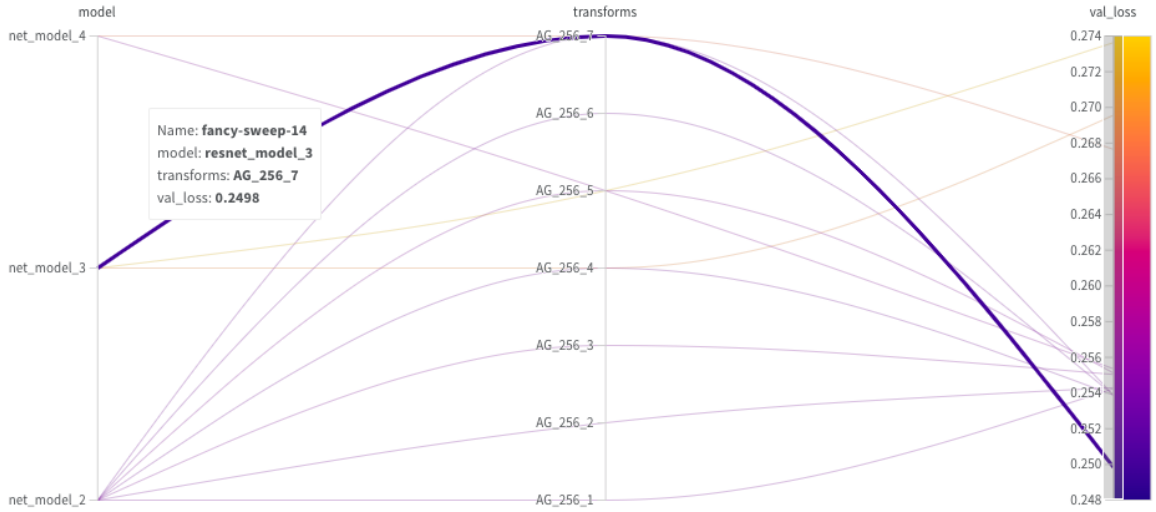


Tabla de metricas:

Transformacion	AG_256_1		AG_256_2		AG_256_3		AG_256_4		AG_256_5		AG_256_6		AG_256_7	
Modelo	2 - LeNet		2 - LeNet		2 - LeNet		2 - LeNet		2 - LeNet		2 - LeNet		2 - LeNet	
clases	0	1	0	1	0	1	2	3	0	1	0	1	0	1
Precision	0,79	0	0,79	0	0,79	0	0,79	0	0,79	0	0,79	0	0,79	0
Recall	1	0	1	0	1	0	1	0	1	0	1	0	1	0
f1-score	0,89	0	0,89	0	0,89	0	0,89	0	0,89	0	0,89	0	0,89	0
support	93	24	93	24	93	24	93	24	93	24	93	24	93	24
Transformacion	AG_256_1		AG_256_2		AG_256_3		AG_256_4		AG_256_5		AG_256_6		AG_256_7	
Modelo	3 - ResNet		3 - ResNet		3 - ResNet		3 - ResNet		3 - ResNet		3 - ResNet		3 - ResNet	
clases	0	1	0	1	0	1	0	1	0	1	0	1	0	1
Precision	0,79	0	0,79	0	0,79	0	0,79	0	0,79	0	0,79	0	0,79	0
Recall	1	0	1	0	1	0	1	0	1	0	1	0	1	0
f1-score	0,89	0	0,89	0	0,89	0	0,89	0	0,89	0	0,89	0	0,89	0
support	93	24	93	24	93	24	93	24	93	24	93	24	93	24
Transformacion	AG_256_1		AG_256_2		AG_256_3		AG_256_4		AG_256_5		AG_256_6		AG_256_7	
Modelo	4 - ResNet		4 - ResNet		4 - ResNet		4 - ResNet		4 - ResNet		4 - ResNet		4 - ResNet	
clases	0	1	0	1	0	1	0	1	0	1	0	1	0	1
Precision	0,79	0	0.80	0,33	0,79	0	0,82	0,29	0,79	0	0,79	0	0,79	0
Recall	1	0	0,98	0,04	1	0	0,82	0,29	1	0	1	0	1	0
f1-score	0,89	0	0,88	0,07	0,89	0	0,82	0,29	0,89	0	0,89	0	0,89	0
support	93	24	93	24	93	24	93	24	93	24	93	24	93	24

Conclusiones de la Tabla: Los valores de la tabla, no se coincide exactamente con lo visto en el grafico, es decir, no se obtiene la mejor adaptabilidad para el modelo 3 con la transformación AG_256_7, sino, que aparenta ser para el modelo 4 transformación AG_256_4.

5.2.2.3 Conclusiones del Experimento 2

En ambos casos analizados, se confirma que una mayor complejidad en la arquitectura de la red mejora su capacidad de adaptación.

Además, se evidencia que la incorporación de data augmentation contribuye positivamente al desempeño, en comparación con los resultados observados en el Experimento 1.

Si bien los resultados aún no son óptimos, ofrecen una base sólida para definir los próximos pasos en el Experimento 3, enfocado en la optimización del modelo.

5.2.3 EXPERIMENTO 3 - Hiperparametros

A partir de los resultados obtenidos en el Experimento 2, se seleccionaron los siguientes modelos para evaluar en profundidad:

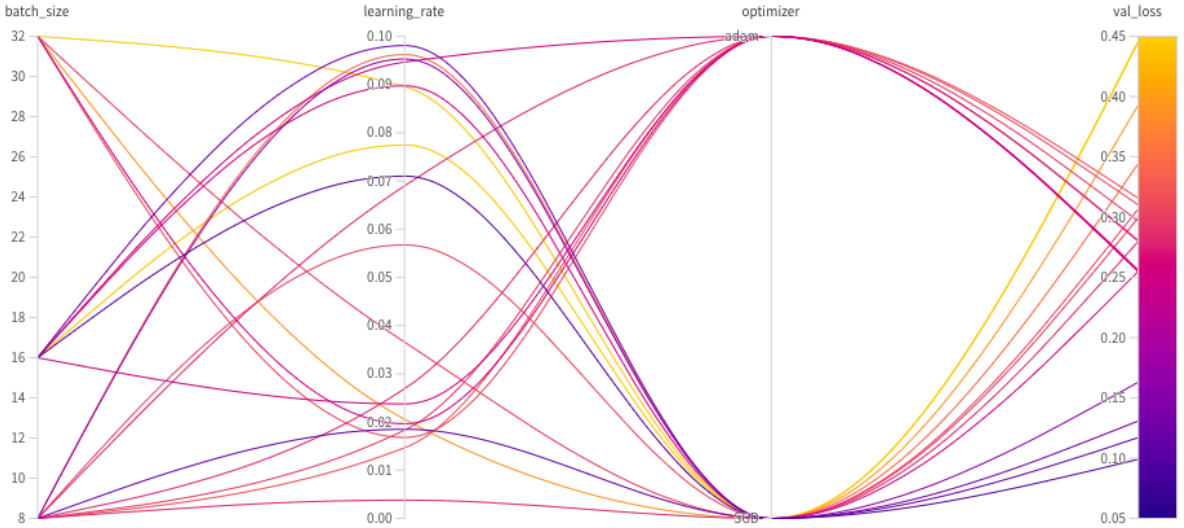
- Modelo 4 con imágenes AG_500_5
- Modelo 4 con imágenes AG_256_4
- Modelo 3 con imágenes AG_256_7

En este experimento, se explorará la optimización de hiperparámetros clave. Los valores a ajustar son:

- Tasa de aprendizaje (LR): rango entre 0.1 y valores cercanos a 0 (seleccionados automáticamente por W&B)
- Tamaño de batch (BATCH_SIZE): [8, 16, 32]
- Optimizador: ['Adam', 'SGD']

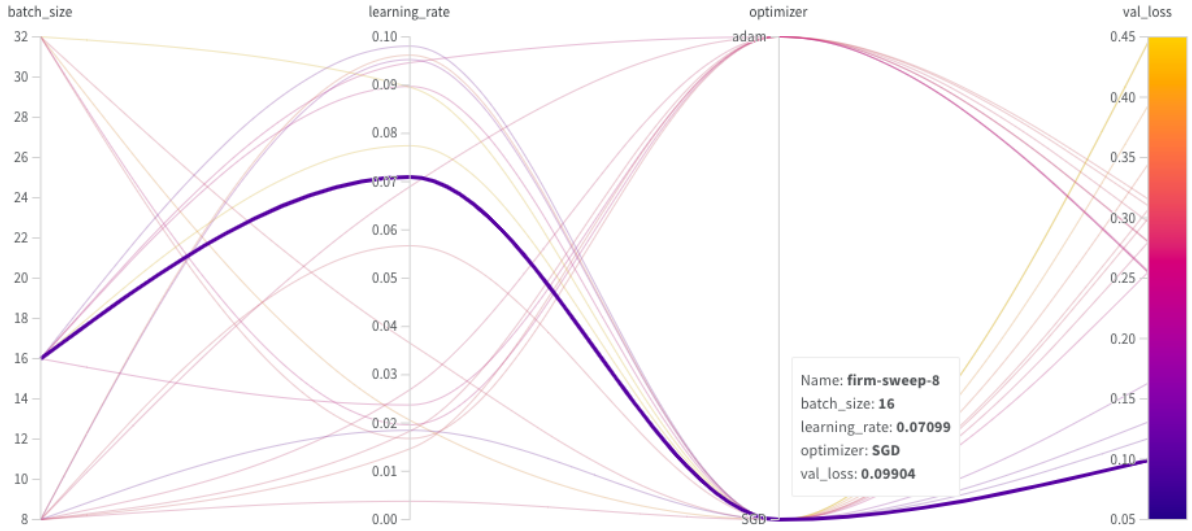
5.2.3.1 Imagenes de 500x500

Gráfico obtenido de W&B para el experimento 2 con imagenes de 500x500



Lo primero que se destaca es que el optimizador SGD muestra un mejor desempeño en comparación con Adam.

A continuación, se identifica la curva con el menor valor de pérdida durante el entrenamiento, la cual se ilustra a continuación:



Se analizan las tablas de resultados:

		Clase	Precision	Recall	f1-score	support
Modelo	4 - ResNet	0	0,79	1	0,89	93
BATCH	8					
LR	0,09620022	1	0	0	0	24
Optimizador	SGD					
Modelo	4 - ResNet	0	0,84	1	0,91	93
BATCH	16					
LR	0,08971782	1	1	0,25	0,4	24
Optimizador	SGD					
Modelo	4 - ResNet	0	0,86	1	0,93	93
BATCH	8					
LR	0,09527858	1	1	0,38	0,55	24
Optimizador	SGD					
Modelo	4 - ResNet	0	0,79	1	0,89	93
BATCH	8					
LR	0,01466349	1	0	0	0	24
Optimizador	Adam					
Modelo	4 - ResNet	0	0,79	1	0,89	93
BATCH	8					
LR	0,05672147	1	0	0	0	24
Optimizador	SGD					
Modelo	4 - ResNet	0	0,79	1	0,89	93
BATCH	32					
LR	0,03656757	1	0	0	0	24
Optimizador	SGD					
Modelo	4 - ResNet	0	0,79	1	0,89	93
BATCH	8					
LR	0,00378816	1	0	0	0	24
Optimizador	SGD					
Modelo	4 - ResNet	0	0,87	1	0,93	93
BATCH	16					
LR	0,07099163	1	1	0,42	0,59	24
Optimizador	SGD					
Modelo	4 - ResNet	0	0,88	0,99	0,93	93
BATCH	16					

LR	0,0980752	1	0,92	0,46	0,61	24
Optimizador	SGD					
Modelo	4 - ResNet	0	0,79	1	0,89	93
BATCH	8					
LR	0,02691047	1	0	0	0	24
Optimizador	Adam					
Modelo	4 - ResNet	0	0,79	1	0,89	93
BATCH	32					
LR	0,0168034	1	0	0	0	24
Optimizador	Adam					
Modelo	4 - ResNet	0	0,79	1	0,89	93
BATCH	16					
LR	0,02375466	1	0	0	0	24
Optimizador	Adam					
Modelo	4 - ResNet	0	0,79	1	0,89	93
BATCH	32					
LR	0,02057862	1	0	0	0	24
Optimizador	SGD					
Modelo	4 - ResNet	0	0,89	1	0,94	93
BATCH	8					
LR	0,01848974	1	1	0,5	0,67	24
Optimizador	SGD					
Modelo	4 - ResNet	0	0,79	1	0,89	93
BATCH	8					
LR	0,01832329	1	0	0	0	24
Optimizador	Adam					
Modelo	4 - ResNet	0	0,79	1	0,89	93
BATCH	16					
LR	0,09464228	1	0	0	0	24
Optimizador	Adam					
Modelo	4 - ResNet	0	0,79	1	0,89	93
BATCH	8					
LR	0,068966	1	0	0	0	24
Optimizador	Adam					
Modelo	4 - ResNet	0	0,79	1	0,89	93
BATCH	32					
LR	0,08957598	1	0	0	0	24

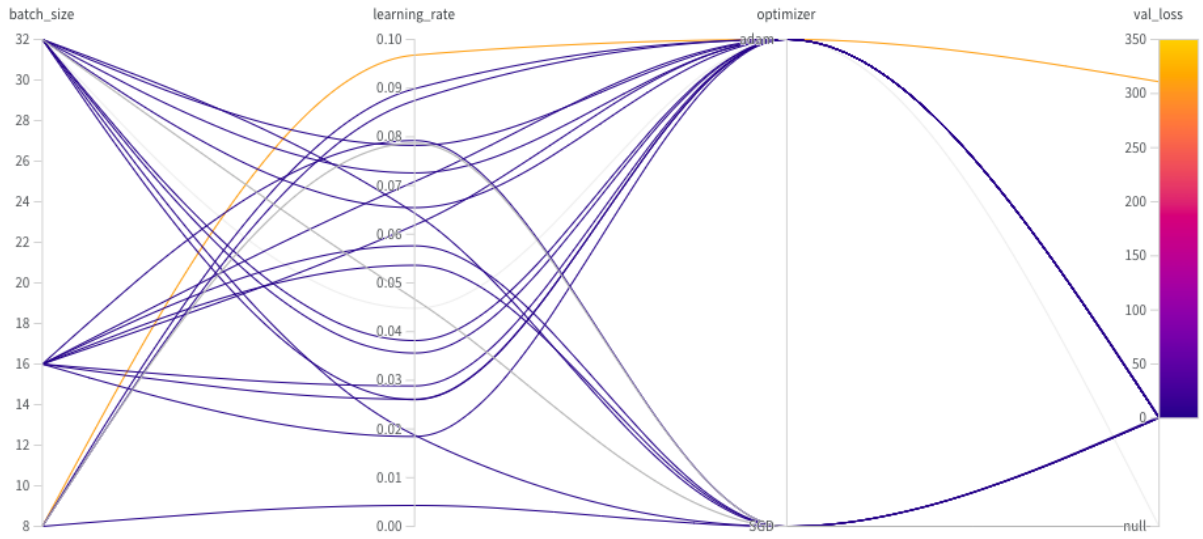
Optimizador	SGD					
Modelo	4 - ResNet	0	0,79	1	0,89	93
BATCH	32					
LR	0,01963289	1	0	0	0	24
Optimizador	Adam					
Modelo	4 - ResNet	0	0,81	0,92	0,86	93
BATCH	16					
LR	0,07744324	1	0,36	0,17	0,23	24
Optimizador	SGD					

Se observa una mejora sustancial en relación con los resultados del Experimento 2.

Si bien aún hay margen para seguir ajustando los hiperparámetros, los valores obtenidos en la corrida con mejor desempeño —identificada previamente en los gráficos— se reflejan también en la tabla, mostrando resultados prometedores.

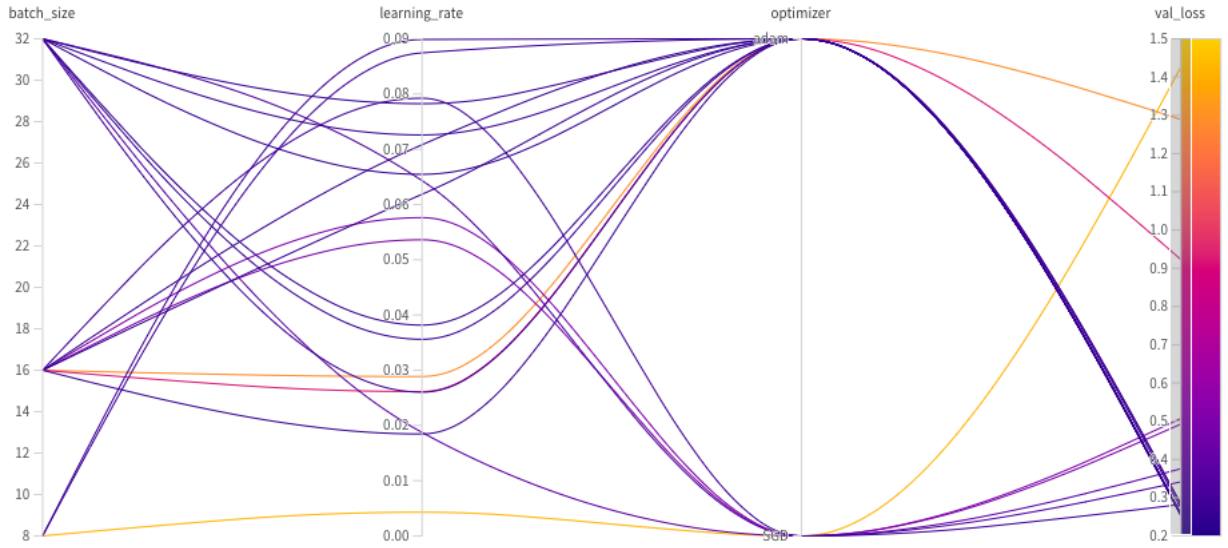
5.2.3.2 Imagenes de 256x256

Se presentan los resultados para el Modelo 4 con imágenes AG_256_4

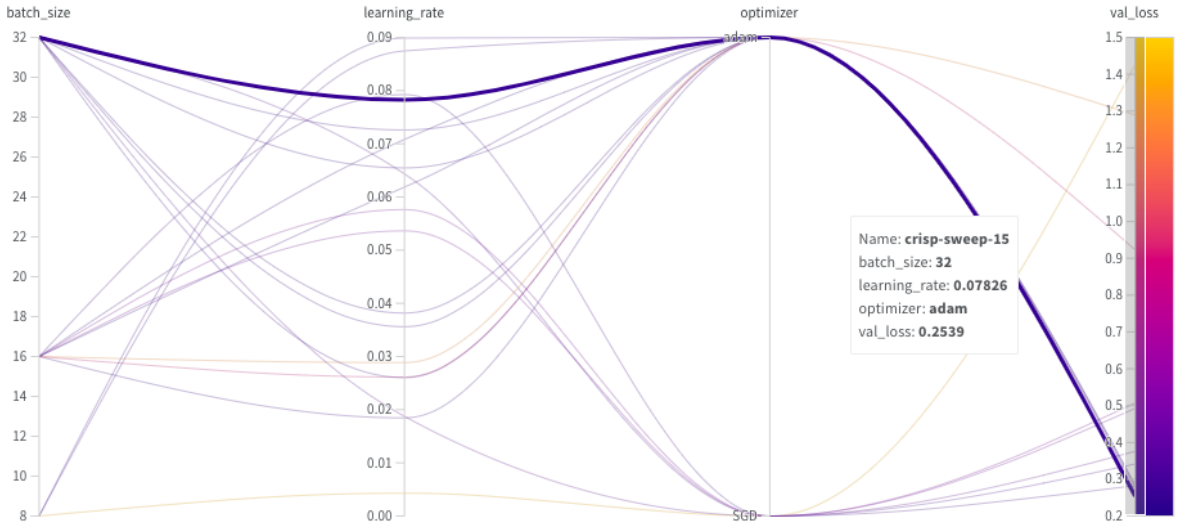


Dado que en esta corrida se obtuvo un valor de pérdida muy elevado, no es posible identificar a simple vista qué optimizador generó mejores resultados.

Para facilitar el análisis, se aplica el siguiente filtro con el objetivo de mejorar la visualización de las curvas más relevantes:



Ahora se observa resultados relativamente parejos en cuanto al optimizador, dando mejores valores adam. En la siguiente grafica se ilustra claramente la mejor corrida:



Tras aplicar el filtro, se observa un rendimiento relativamente parejo entre los optimizadores, aunque Adam muestra una leve ventaja en los mejores valores obtenidos.

En la siguiente gráfica se ilustra con claridad la mejor corrida del experimento.

Se continua con la tabla de metricas:

		Clase	Precision	Recall	f1-score	support
Modelo	4 - ResNet	0	0,79	1	0,89	93
BATCH	16					
LR	0,05361845	1	0	0	0	24
Optimizador	SGD					
Modelo	4 - ResNet	0	0,79	1	0,89	93
BATCH	16					
LR	0,01846183	1	0	0	0	24
Optimizador	Adam					
Modelo	4 - ResNet	0	0,79	1	0,89	93
BATCH	32					
LR	0,06546531	1	0	0	0	24
Optimizador	Adam					
Modelo	4 - ResNet	0	0,79	1	0,89	93
BATCH	8					
LR	0,08745103	1	0	0	0	24
Optimizador	Adam					
Modelo	4 - ResNet	0	0,79	1	0,89	93
BATCH	16					
LR	0,07081167	1	0	0	0	24
Optimizador	Adam					
Modelo	4 - ResNet	0	0,81	0,97	0,88	93
BATCH	32					
LR	0,06433135	1	0,5	0,12	0,2	24
Optimizador	SGD					
Modelo	4 - ResNet	0	0,79	1	0,89	93
BATCH	8					
LR	0,0967841	1	0	0	0	24
Optimizador	Adam					
Modelo	4 - ResNet	0	0,79	1	0,89	93
BATCH	32					

LR	0,01874516	1	0	0	0	24
Optimizador	SGD					
Modelo	4 - ResNet	0	0,79	1	0,89	93
BATCH	16					
LR	0,02613252	1	0	0	0	24
Optimizador	Adam					
Modelo	4 - ResNet	0	0,79	1	0,89	93
BATCH	16					
LR	0,06185099	1	0	0	0	24
Optimizador	Adam					
Modelo	4 - ResNet	0	0,79	1	0,89	93
BATCH	32					
LR	0,03817224	1	0	0	0	24
Optimizador	Adam					
Modelo	4 - ResNet	0	0,79	1	0,89	93
BATCH	32					
LR	0,07825751	1	0	0	0	24
Optimizador	Adam					
Modelo	4 - ResNet	0	0,79	1	0,89	93
BATCH	16					
LR	0.07923592776578908	1	0	0	0	24
Optimizador	SGD					
Modelo	4 - ResNet	0	0,85	0,42	0,56	93
BATCH	8					
LR	0,00430031	1	1	0,71	0,36	24
Optimizador	SGD					
Modelo	4 - ResNet	0	1	0,01	0,02	93
BATCH	16					
LR	0,028859	1	0,21	1	0,34	24
Optimizador	Adam					
Modelo	4 - ResNet	0	0,79	1	0,89	93
BATCH	32					

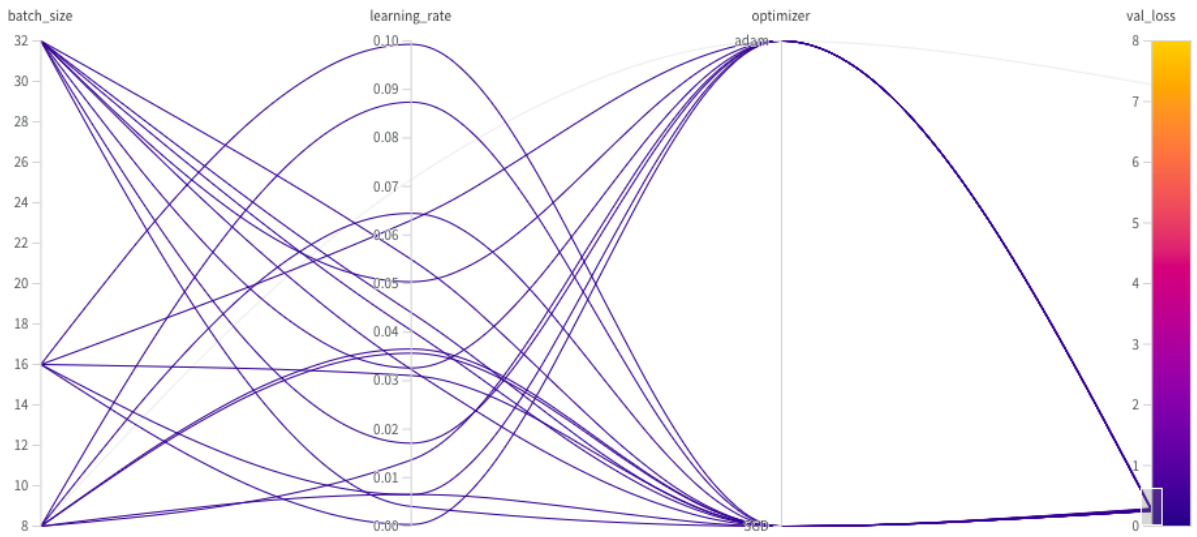
LR	0,02602613	1	0	0	0	24
Optimizador	Adam					
Modelo	4 - ResNet	0	0,79	1	0,89	93
BATCH	8					
LR	0,08987406	1	0	0	0	24
Optimizador	Adam					
Modelo	4 - ResNet	0	0,79	1	0,89	93
BATCH	32					
LR	0,0725957	1	0	0	0	24
Optimizador	Adam					
Modelo	4 - ResNet	0	0,79	1	0,89	93
BATCH	32					
LR	0,03560497	1	0	0	0	24
Optimizador	Adam					
Modelo	4 - ResNet	0	0,79	1	0,89	93
BATCH	16					
LR	0,05761856	1	0	0	0	24
Optimizador	SGD					

En este caso, los resultados obtenidos no coinciden con lo observado en el gráfico: el menor valor de la función de pérdida en el conjunto de validación no se traduce en un buen desempeño en la tabla de métricas.

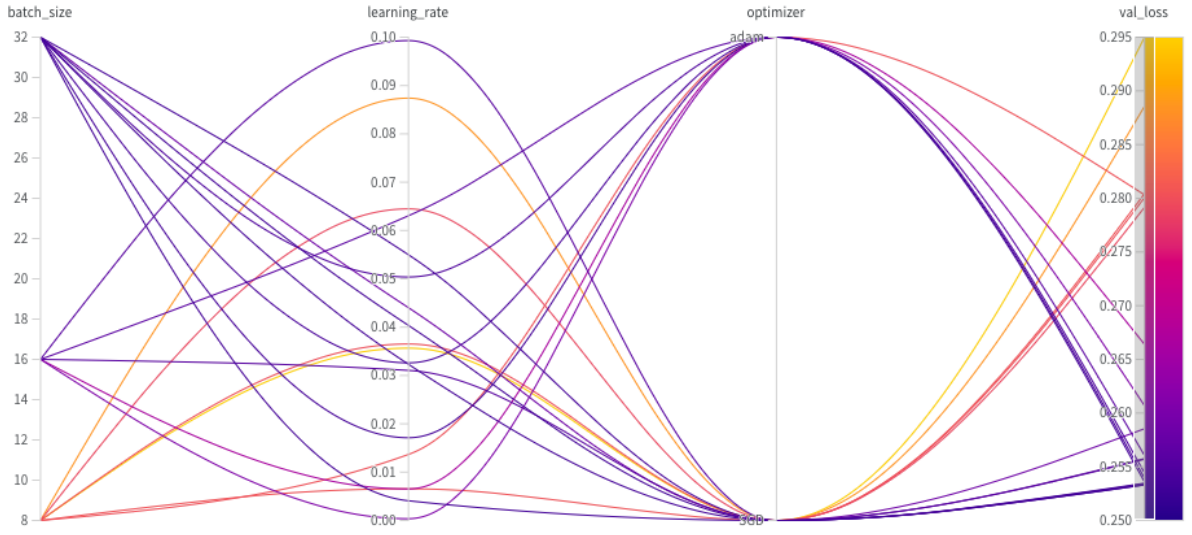
Esto sugiere que el modelo no logró captar correctamente la distribución de los datos, reflejándose en una precisión habitual de 0.79 para la clase 1 y 0 para la clase 0.

Se continúa con la evaluación del siguiente modelo previsto para esta dimensión.

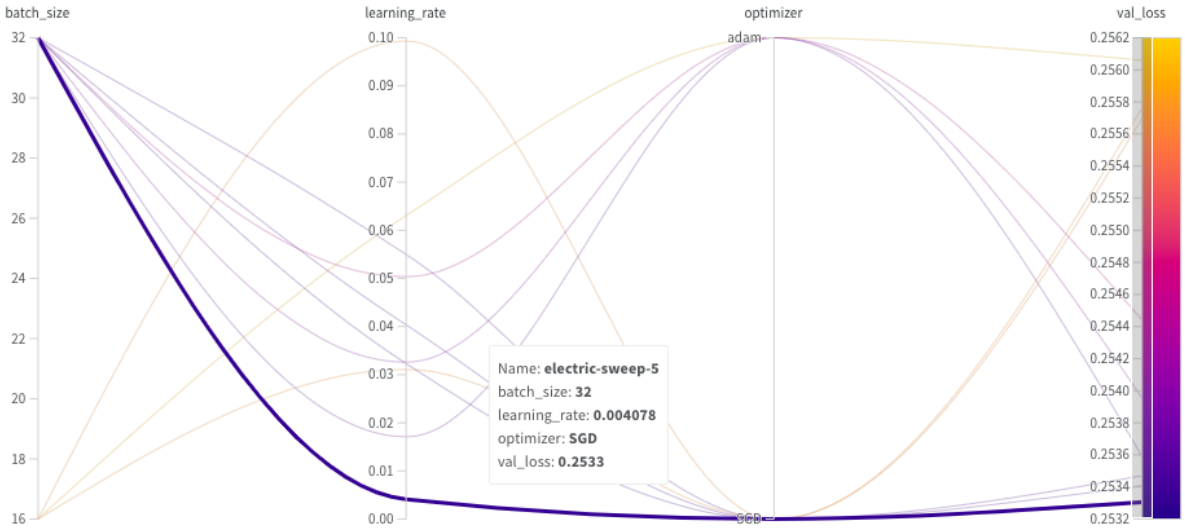
Modelo 3 con imágenes AG_256_7



Conclusión del grafico: Nuevamente, debido a un valor mayor, los resultados que interesan quedan fuera de escala y es preciso aplicar un filtro para mejorar la visualización:



En este caso, se observan buenos resultados para ambos optimizadores. Para facilitar una comparación más precisa entre las mejores corridas, se aplica un segundo filtro que mejora la visualización de las curvas más relevantes.



Con el segundo filtro aplicado, se aprecia con claridad la curva correspondiente a la mejor performance del experimento en este caso.

Se procede a observar la tabla de resultados de metrics:

		Clase	Precision	Recall	f1-score	support
Modelo	3 - ResNet	0	0,79	1	0,89	93
BATCH	8					
LR	0,01366686	1	0	0	0	24
Optimizador	Adam					
Modelo	3 - ResNet	0	0,79	1	0,89	93
BATCH	32					
LR	0,04402525	1	0	0	0	24
Optimizador	SGD					
Modelo	3 - ResNet	0	0,79	1	0,89	93
BATCH	8					
LR	0,03651541	1	0	0	0	24
Optimizador	SGD					
Modelo	3 - ResNet	0	0,79	1	0,89	93
BATCH	8					
LR	0,0644748	1	0	0	0	24
Optimizador	SGD					
Modelo	3 - ResNet	0	0,79	1	0,89	93

BATCH	32					
LR	0,0040777	1	0	0	0	24
Optimizador	SGD					
Modelo	3 - ResNet	0	0,79	1	0,89	93
BATCH	8					
LR	0,03565077	1	0	0	0	24
Optimizador	SGD					
Modelo	3 - ResNet	0	0,79	1	0,89	93
BATCH	32					
LR	0,03261922	1	0	0	0	24
Optimizador	Adam					
Modelo	3 - ResNet	0	0,79	1	0,89	93
BATCH	8					
LR	0,00655016	1	0	0	0	24
Optimizador	SGD					
Modelo	3 - ResNet	0	0,79	1	0,89	93
BATCH	32					
LR	0,03231538	1	0	0	0	24
Optimizador	Adam					
Modelo	3 - ResNet	0	0,79	1	0,89	93
BATCH	16					
LR	0,00652751	1	0	0	0	24
Optimizador	Adam					
Modelo	3 - ResNet	0	0,79	1	0,89	93
BATCH	16					
LR	0,06305815	1	0	0	0	24
Optimizador	Adam					
Modelo	3 - ResNet	0	0,79	1	0,89	93
BATCH	32					
LR	0,01712008	1	0	0	0	24
Optimizador	Adam					
Modelo	3 - ResNet	0	0,79	1	0,89	93
BATCH	32					
LR	0,0503841	1	0	0	0	24
Optimizador	Adam					
Modelo	3 - ResNet	0	0,79	1	0,89	93
BATCH	16					

LR	0,03105127	1	0	0	0	24
Optimizador	SGD					
Modelo	3 - ResNet	0	0,79	1	0,89	93
BATCH	8					
LR	0,08738981	1	0	0	0	24
Optimizador	SGD					
Modelo	3 - ResNet	0	0,79	1	0,89	93
BATCH	8					
LR	0,07144701	1	0	0	0	24
Optimizador	Adam					
Modelo	3 - ResNet	0	0,79	1	0,89	93
BATCH	16					
LR	0,09930004	1	0	0	0	24
Optimizador	SGD					
Modelo	3 - ResNet	0	0,79	1	0,89	93
BATCH	32					
LR	0,05497779	1	0	0	0	24
Optimizador	SGD					
Modelo	3 - ResNet	0	0,79	1	0,89	93
BATCH	16					
LR	0,00036678	1	0	0	0	24
Optimizador	Adam					
Modelo	3 - ResNet	0	0,79	1	0,89	93
BATCH	32					
LR	0,04036049	1	0	0	0	24
Optimizador	SGD					

Este es otro caso en el que el mejor resultado observado en el gráfico no se refleja en las métricas de la tabla.

Esto confirma que, para este modelo y estas dimensiones, incluso con ajuste de hiperparámetros, no se logra una respuesta satisfactoria.

5.2.3.3 Conclusiones del experimento 3

En base a los resultados obtenidos, el mejor desempeño corresponde al Modelo 4 (ResNet) con la transformación AG_500_5.

Esta configuración logró el mejor equilibrio entre la pérdida y las métricas de clasificación, destacándose frente al resto de las variantes evaluadas. Además, fue el único caso en el que los resultados de las métricas se correlacionaron claramente con la evolución de la función de pérdida sobre el conjunto de validación.

5.2.4 EXPERIMENTO 4 - Ajuste fino de hiperparametros

A partir de los resultados obtenidos en los experimentos anteriores, se definió trabajar con la siguiente configuración base:

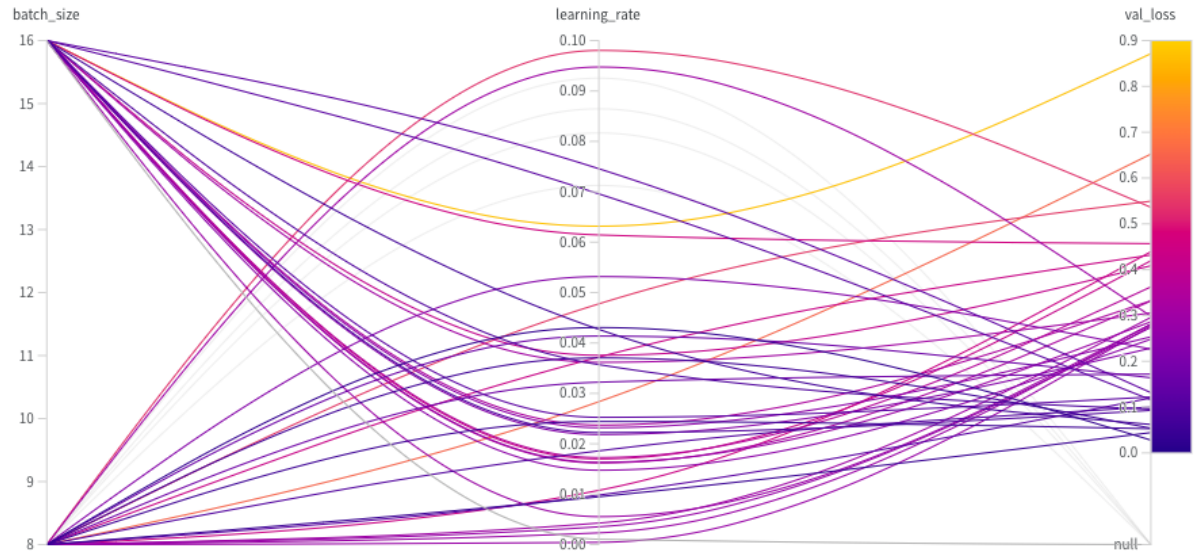
- Modelo: 4 (ResNet)
- Transformación: AG_500_5
- Dimensión de imagen: 500×500 píxeles
- Optimizador: SGD
- Épocas: 30

En esta etapa, el objetivo es realizar un ajuste fino de los hiperparámetros restantes. Para ello, se exploran las siguientes combinaciones:

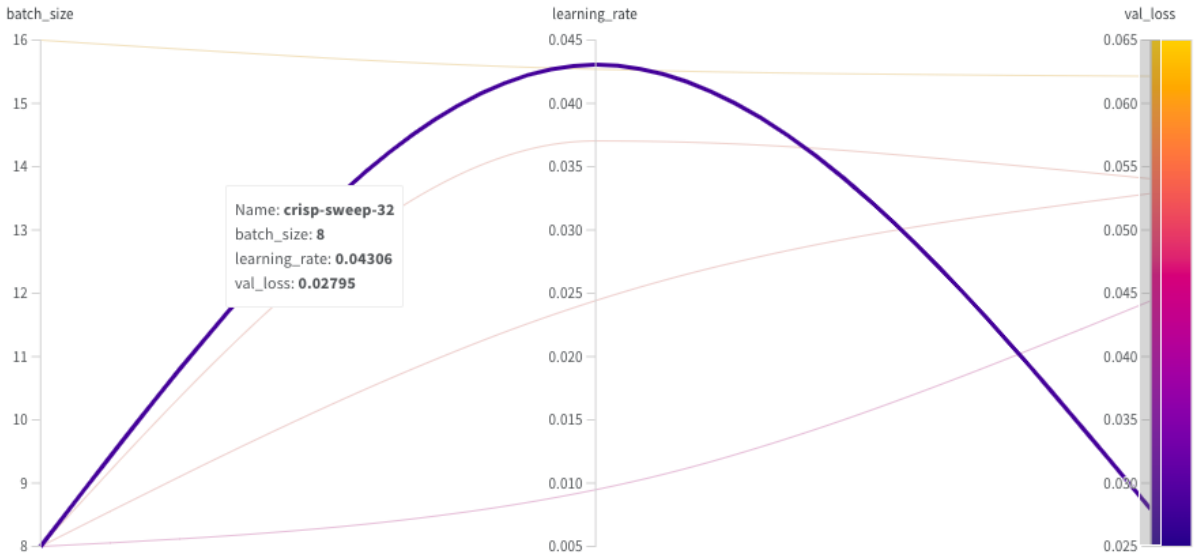
- Tasa de aprendizaje (LR): valores en el rango entre 0.1 y cercanos a 0, seleccionados automáticamente por W&B
- Tamaño de batch (BATCH_SIZE): [8, 16]

5.2.4.1 Resultados

Se ilustran los resultados a continuacion:



Se ajusta la escala para mejor visualizacion



Se procede a revisar las métricas para evaluar conclusiones:

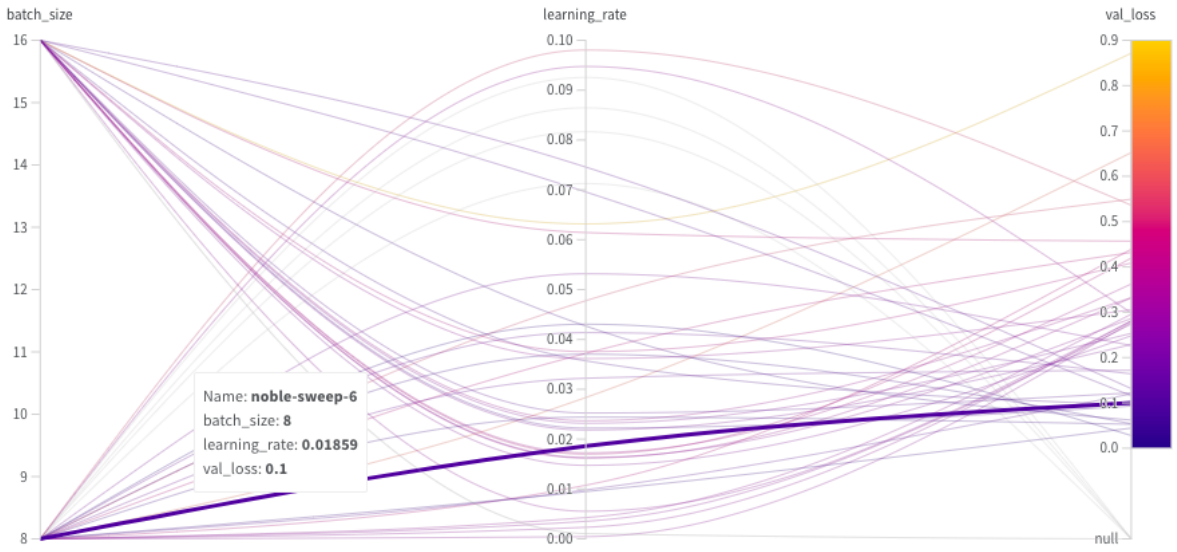
		Clase	Precision	Recall	f1-score	support
BATCH	8	0	0,79	1	0,89	93
LR	0,00232566	1	0	0	0	24
BATCH	8	0	0,86	1	0,93	93
LR	0,04136571	1	1	0,38	0,55	24
BATCH	8	0	0,97	0,98	0,97	93
LR	0,01859257	1	0,91	0,88	0,89	24
BATCH	16	0	0,79	1	0,89	93
LR	0,06143858	1	0	0	0	24
BATCH	16	0	0,79	1	0,89	93
LR	0,00352891	1	0	0	0	24
BATCH	8	0	0,79	1	0,89	93
LR	0,01076209	1	0	0	0	24
BATCH	16	0	0,83	1	0,91	93
LR	0,01475513	1	1	0,21	0,34	24
BATCH	16	0	0,8	1	0,89	93
LR	0,00554276	1	1	0,04	0,08	24
BATCH	16	0	0,79	1	0,89	93
LR	0,01699149	1	0	0	0	24
BATCH	16	0	0,82	1	0,9	93
LR	0,02183839	1	1	0,12	0,22	24
BATCH	8	0	0,79	1	0,89	93
LR	0,03720798	1	0	0	0	24
BATCH	8	0	0,84	1	0,91	93
LR	0,03221736	1	1	0,25	0,4	24
BATCH	16	0	0,89	1	0,94	93
LR	0,03570752	1	1	0,5	0,67	24
BATCH	16	0	0,85	1	0,92	93
LR	0,02525308	1	1	0,33	0,5	24
BATCH	16	0	0,9	1	0,95	93
LR	0,04269765	1	1	0,58	0,74	24
BATCH	16	0	0,79	1	0,89	93
LR	0,03756276	1	0	0	0	24
BATCH	16	0	0,79	1	0,89	93
LR	0,06317823	1	0	0	0	24
BATCH	8	0	0,91	0,99	0,95	93

LR	0,00948199	1	0,94	0,62	0,75	24
BATCH	8	0	0,79	1	0,89	93
LR	0,00427309	1	0	0	0	24
BATCH	16	0	0,88	1	0,93	93
LR	0,02223145	1	1	0,46	0,63	24
BATCH	16	0	0,8	1	0,89	93
LR	0,01726587	1	1	0,04	0,08	24
BATCH	16	0	0,8	1	0,89	93
LR	0,0231918	1	1	0,04	0,08	24
BATCH	8	0	0,79	1	0,89	93
LR	0,00044489	1	0	0	0	24
BATCH	8	0	0,79	1	0,89	93
LR	0,04776415	1	0	0	0	24
BATCH	16	0	0,79	1	0,89	93
LR	0,02370662	1	0	0	0	24
BATCH	8	0	0,97	1	0,98	93
LR	0,04305762	1	1	0,88	0,93	24
BATCH	8	0	0,79	1	0,89	93
LR	0,09474713	1	0	0	0	24
BATCH	16	0	0,82	0,99	0,9	93
LR	0,01620292	1	0,8	0,17	0,28	24
BATCH	8	0	0,79	1	0,89	93
LR	0,02835285	1	0	0	0	24
BATCH	8	0	0,92	0,99	0,95	93
LR	0,03703533	1	0,94	0,67	0,78	24
BATCH	8	0	0,95	1	0,97	93
LR	0,02441971	1	1	0,79	0,88	24
BATCH	8	0	0,79	1	0,89	93
LR	0,09804134	1	0	0	0	24
BATCH	8	0	0,82	1	0,9	93
LR	0,05316766	1	1	0,17	0,29	24
BATCH	16	0	0,79	1	0,89	93
LR	0,03616565	1	0	0	0	24
BATCH	16	0	0,87	1	0,93	93
LR	0,06958267	1	1	0,42	0,59	24
BATCH	16	0	0,87	1	0,93	93
LR	0,07457448	1	1	0,42	0,59	24

BATCH	16	0	0,79	1	0,89	93
LR	0,01633488	1	0	0	0	24

Conclusiones

En este caso, el mejor grafico segun el valor de la funcion de perdida no fue el mejor resultado de las metricas. El mejor fue el correspondiente a nobel-sweep. A continuacion se resume los resultados para este experimento:



		Clase	Precision	Recall	f1-score	support
BATCH	8	0	0,97	0,98	0,97	93
LR	0,01859257	1	0,91	0,88	0,89	24

Se selecciona esta corrida por presentar el mejor equilibrio en las métricas, especialmente en el valor de F1-score.

Si bien el gráfico de loss para entrenamiento y validación no muestra una tendencia perfectamente estable, ambos presentan una disminución general, lo que indica un proceso de aprendizaje adecuado. Esto refuerza la elección de esta configuración como la más prometedora en esta etapa del experimento.

6. API de Predicción

6.1 Implementación

Framework: FastAPI Características:

- Predicciones online (tiempo real)
- Predicciones batch (procesamiento por lotes)
- Documentación automática con OpenAPI
- Validación de entrada con Pydantic

Se crea un README especifio para poder aplicarcarlo y correrlo en cualquier lado, desplegandolo local o con Docker.

https://github.com/valeriaeskenazi/Obligatorio_ML/blob/main/api/README-API.md

6.2 Endpoints

```
POST /predict/single    # Predicción individual
POST /predict/batch     # Predicciones en lote
GET /health             # Estado del servicio
GET /docs               # Documentación interactiva
```

6.3 Formato de Respuesta

```
{
   "prediction": "octagon_detected",
   "confidence": 0.95,
   "processing_time": 0.234
}
```

7. Requerimientos Electivos Implementados

7.1 Trazabilidad de ML

Herramientas: Weights & Biases (wandb) Elementos versionados:

- Experimentos: Tracking completo en wandb con hiperparámetros, métricas y artefactos
- Modelos: Versionado automático en wandb con metadatos
- Datos: Organización en S3 con estructura de carpetas por versión

Beneficios:

- Reproducibilidad completa de experimentos
- Comparación sistemática de modelos
- Auditoría de cambios en datos

7.2 Optimización de Modelos

7.2.1 Ajuste de hiperparámetros

Durante la etapa del Experimento 4, se llevó a cabo un ajuste fino de los hiperparámetros utilizando la herramienta Weights & Biases (W&B). Esta exploración permitió alcanzar resultados satisfactorios, especialmente si se los compara con las corridas iniciales y teniendo en cuenta el desbalance de clases presente en el conjunto de datos.

Los resultados finales del modelo ajustado se detallan a continuación:

Clase	Precision	Recall	f1-score	support
0	0,97	0,98	0,97	93
1	0,91	0,88	0,89	24

7.2.2 Pruning

Dado que se trabaja con imágenes de alta resolución y una arquitectura de red profunda, se aplicó la técnica de pruning para reducir la complejidad del modelo.

Se propuso una reducción del 30% de los pesos, lo que resultó en una caída del 2,56% en la Accuracy general (de 95,73% a 93,16%).

7.2.3 Fine-tuning

Para compensar la pérdida de Accuracy ocasionada por el pruning, se realizó un proceso de fine-tuning durante 3 épocas. Como resultado, se logró no solo recuperar el desempeño original, sino incluso superarlo levemente.

A continuación, se resumen las métricas obtenidas tras el fine-tuning:

Clase	Precision	Recall	f1-score	support
0	0,97	0,99	0,98	93
1	0,99	0,88	0,91	24

7.3 Visualización

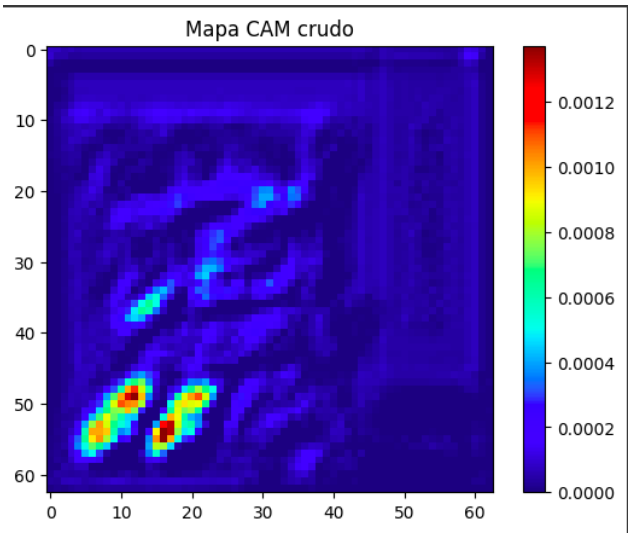
Gradio fue la herramienta elegida para crear una interfaz de usuario a través de la cual interactuar con el modelo. Se puede subir una simple imagen, o se pueden subir varias imágenes en simultáneo, dependiendo si se quiere hacer una predicción online, o se quiere probar el endpoint para predicción en batch. En la parte superior también se puede realizar una llamada al endpoint que verifica la salud del servidor.

7.4 Explicabilidad

Se implementó un código basado en Grad-CAM para evaluar la explicabilidad del modelo mediante mapas de calor.

En las imágenes presentadas a continuación, se observa que la red está interpretando correctamente el contenido visual: los focos de activación se concentran sobre los octógonos, que son el objetivo principal del clasificador.

Esto evidencia que el modelo no solo acierta en su predicción, sino que lo hace basándose en regiones relevantes de la imagen, lo cual refuerza la confianza en su funcionamiento.



8. Resultados y Evaluación

8.1 Métricas de Rendimiento

Tomando el promedio para las dos clases el resultado final fue:

Métrica	Valor
Accuracy	96,58%
Precision	96%
Recall	93%
F1-Score	97%

8.3 Casos de Uso y Limitaciones

Fortalezas:

- Alta precisión en imágenes de buena calidad
- Procesamiento eficiente en tiempo real
- Arquitectura escalable

Limitaciones identificadas:

- Sensibilidad a calidad de imagen
- Dependencia de iluminación
- Generalización a otros tipos de etiquetado y fondos

9. Despliegue y Operación

9.1 Estrategia de Despliegue

Entorno: AWS EC2 con contenedores Docker Escalabilidad: Auto-scaling basado en métricas de uso Monitoreo: CloudWatch para métricas de sistema y aplicación

9.2 Consideraciones de Producción

Seguridad:

- Autenticación API mediante tokens
- Validación de entrada robusta
- Logging de seguridad

En el README_Nueva_Version, se detalla brevemente las instrucciones realizadas para poder realizar el despligue

Enlace en:

https://github.com/valeriaeskenazi/Obligatorio_ML/blob/main/README_Nueva_version.md

10. Conclusiones y Trabajo Futuro

10.1 Logros Principales

- Sistema end-to-end completamente funcional
- Integración exitosa de múltiples tecnologías
- Despliegue exitoso en producción

10.2 Aprendizajes Clave

- Importancia de la calidad de datos en el rendimiento
- Valor de la automatización en el pipeline de ML
- Necesidad de monitoreo continuo en producción

10.3 Mejoras Futuras

- Implementación de feedback loop para mejora continua
- Expansión a múltiples tipos de etiquetado
- Aplimiacion del datos de entrenmiento para mayor adaptabilidad (por ejemplo, frente a imágenes con fondos)

11. Referencias

1. Huyen, C. (2022). Diseño de sistemas de machine learning. O'Reilly Media.
2. (Mayr, Machine Learning en Producción Datos de entrenamiento: Muestreo y repaso, 2025)
3. (Mayr, Machine Learning en Producción Datos: definición y recolección, 2025)
4. (Mayr, Machine Learning en Producción Elección de los modelos, 2025)
5. (Mayr, Machine Learning en Producción Etiquetado de datos, 2025)
6. (Mayr, Machine Learning en Producción Feature Engineering, 2025)
7. (Mayr, Machine Learning en Producción Feature Engineering, 2025)
8. (Mayr, Machine Learning en Producción Introducción, 2025)