Conclusiones Baseline

Integrantes:

- Espínola, Carla
- Gambarte, Antonella Nerea
- Torres, Dimas Ignacio

Metodología

Para seleccionar las arquitecturas que se aplicarán en este trabajo, consideramos tanto el problema que se desea resolver como su posible utilidad en un contexto real. Dado que se trata de un problema de clasificación en el cual se entrena un modelo para identificar una enfermedad específica en una especie de planta, creemos que esta solución podría resultar especialmente útil en aplicaciones móviles dirigidas, por ejemplo, a agricultores, ya que estas aplicaciones les permitirían detectar enfermedades en sus plantaciones en un dispositivo de limitados recursos.

Es por ello que elegimos la arquitectura *MobileNet*, diseñada específicamente para funcionar de manera eficiente en dispositivos móviles gracias a su estructura ligera y de bajo costo computacional.

Para llevar a cabo las primeras pruebas, utilizamos el modelo *MobileNet_V2* de PyTorch, preentrenado en ImageNet. En esta etapa aplicamos únicamente las transformaciones mínimas necesarias. Según la documentación de PyTorch (Link), es necesario normalizar las imágenes con los valores de media y desviación estándar utilizados durante el entrenamiento en ImageNet, que se muestran a continuación:

All pre-trained models expect input images normalized in the same way, i.e. mini-batches of 3-channel RGB images of shape (3 x H x W), where H and W are expected to be at least 224. The images have to be loaded in to a range of [0, 1] and then normalized using `mean = [0.485, 0.456, 0.406]` and `std = [0.229, 0.224, 0.225]`. You can use the following transform to normalize:

```
normalize = transforms.Normalize(mean=[0.485, 0.456, 0.406],
                                 std=[0.229, 0.224, 0.225])
```

Además, ajustamos el tamaño de las imágenes mediante resize a 224×224 píxeles, que corresponde al mínimo requerido.

Teniendo en cuenta los resultados obtenidos durante la etapa de análisis exploratorio de datos , identificamos que era necesario trabajar con imágenes a color, por lo que en esta etapa utilizamos dicho tipo de imágenes.

En cuanto a los pesos disponibles, a continuación se presentan las opciones para MobileNet_V2:

Weight	Acc@1	Acc@5	Params	GFLOPS	Recipe
AlexNet_Weights.IMAGENET1K_V1	56.522	79.066	61.1M	0.71	link
ConvNeXt_Base_Weights.IMAGENET1K_V1	84.062	96.87	88.6M	15.36	link
ConvNeXt_Large_Weights.IMAGENET1K_V1	84.414	96.976	197.8M	34.36	link
ConvNeXt_Small_Weights.IMAGENET1K_V1	83.616	96.65	50.2M	8.68	link
ConvNeXt_Tiny_Weights.IMAGENET1K_V1	82.52	96.146	28.6M	4.46	link
DenseNet121_Weights.IMAGENET1K_V1	74.434	91.972	8.0M	2.83	link
DenseNet161_Weights.IMAGENET1K_V1	77.138	93.56	28.7M	7.73	link
DenseNet169_Weights.IMAGENET1K_V1	75.6	92.806	14.1M	3.36	link
DenseNet201_Weights.IMAGENET1K_V1	76.896	93.37	20.0M	4.29	link
EfficientNet_B0_Weights.IMAGENET1K_V1	77.692	93.532	5.3M	0.39	link
EfficientNet_B1_Weights.IMAGENET1K_V1	78.642	94.186	7.8M	0.69	link
EfficientNet_B1_Weights.IMAGENET1K_V2	79.838	94.934	7.8M	0.69	link
EfficientNet_B2_Weights.IMAGENET1K_V1	80.608	95.31	9.1M	1.09	link
EfficientNet_B3_Weights.IMAGENET1K_V1	82.008	96.054	12.2M	1.83	link
EfficientNet_B4_Weights.IMAGENET1K_V1	83.384	96.594	19.3M	4.39	link
EfficientNet_B5_Weights.IMAGENET1K_V1	83.444	96.628	30.4M	10.27	link
EfficientNet_B6_Weights.IMAGENET1K_V1	84.008	96.916	43.0M	19.07	link
EfficientNet_B7_Weights.IMAGENET1K_V1	84.122	96.908	66.3M	37.75	link
EfficientNet_V2_L_Weights.IMAGENET1K_V1	85.808	97.788	118.5M	56.08	link
EfficientNet_V2_M_Weights.IMAGENET1K_V1	85.112	97.156	54.1M	24.58	link
EfficientNet_V2_S_Weights.IMAGENET1K_V1	84.228	96.878	21.5M	8.37	link
GoogLeNet_Weights.IMAGENET1K_V1	69.778	89.53	6.6M	1.5	link
Inception_V3_Weights.IMAGENET1K_V1	77.294	93.45	27.2M	5.71	link
MNASNet0_5_Weights.IMAGENET1K_V1	67.734	87.49	2.2M	0.1	link
MNASNet0_75_Weights.IMAGENET1K_V1	71.18	90.496	3.2M	0.21	link
MNASNet1_0_Weights.IMAGENET1K_V1	73.456	91.51	4.4M	0.31	link
MNASNet1_3_Weights.IMAGENET1K_V1	76.506	93.522	6.3M	0.53	link
MaxVit_T_Weights.IMAGENET1K_V1	83.7	96.722	30.9M	5.56	link
MobileNet_V2_Weights.IMAGENET1K_V1	71.878	90.286	3.5M	0.3	link
MobileNet_V2_Weights.IMAGENET1K_V2	72.154	90.822	3.5M	0.3	link
MobileNet_V3_Large_Weights.IMAGENET1K_V1	74.042	91.34	5.5M	0.22	link
MobileNet_V3_Large_Weights.IMAGENET1K_V2	75.274	92.566	5.5M	0.22	link
MobileNet_V3_Small_Weights.IMAGENET1K_V1	67.668	87.402	2.5M	0.06	link

En nuestro caso, utilizamos IMAGENET1K_V1. Aunque la diferencia en accuracy entre las versiones disponibles no es sustancial, optamos por la versión con menor precisión para esta etapa baseline. Después se evaluará si existe una mejora o no utilizando otros pesos.

Congelamiento de la red base

Se congelaron los parámetros del feature extractor para mantener los pesos preentrenados y evitar que se modifiquen durante el entrenamiento. De este modo, solo entrenamos la nueva capa clasificadora, lo que reduce el tiempo de entrenamiento.

Adaptación de la capa clasificadora

La capa final del modelo se reemplazó por una nueva secuencia de capas:

Dropout (0.2): ayuda a prevenir el sobreajuste durante el entrenamiento.

Linear Layer: adapta el número de salidas a 38, correspondiente a las clases de enfermedades vegetales presentes en nuestro dataset.

Función de pérdida y optimizador

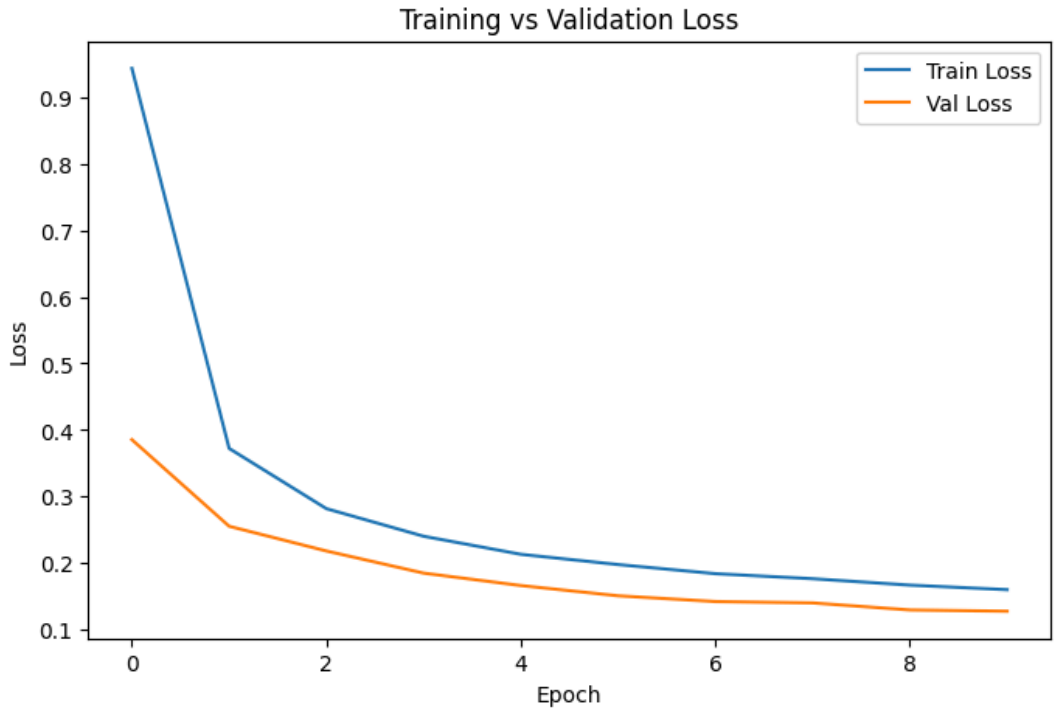
Para entrenar la nueva capa se utilizó la función de pérdida estándar para clasificación multiclase (CrossEntropyLoss). Como optimizador se eligió el SGD, el optimizador más básico, aplicándose únicamente a los parámetros de la nueva capa. En etapas posteriores se analizará la necesidad o no de utilizar optimizadores más avanzados.

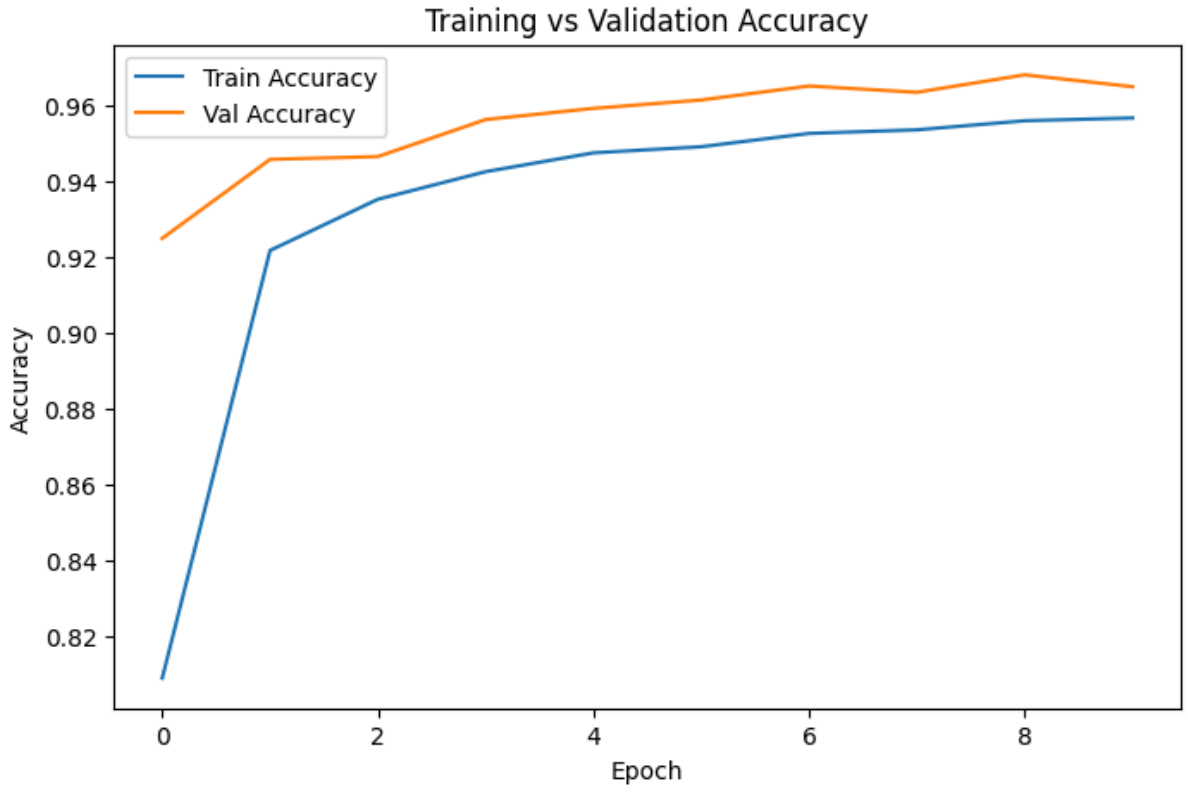
Resultados

1. Entrenamiento y validación

Durante la fase de entrenamiento, se monitorearon las curvas de pérdida (*loss*) y precisión (*accuracy*) tanto para el conjunto de entrenamiento como para el de validación.

- Curvas de Loss y Accuracy: Se observa que el modelo aprende rápidamente en las primeras épocas, alcanzando una alta precisión en el conjunto de entrenamiento. Sin embargo, la brecha entre la precisión de entrenamiento y la de validación, junto con el comportamiento de la curva de pérdida, sugiere que el modelo podría beneficiarse de técnicas de regularización adicionales o *data augmentation* para generalizar mejor, ya que la validación tiende a estabilizarse o fluctuar mientras el entrenamiento sigue mejorando.



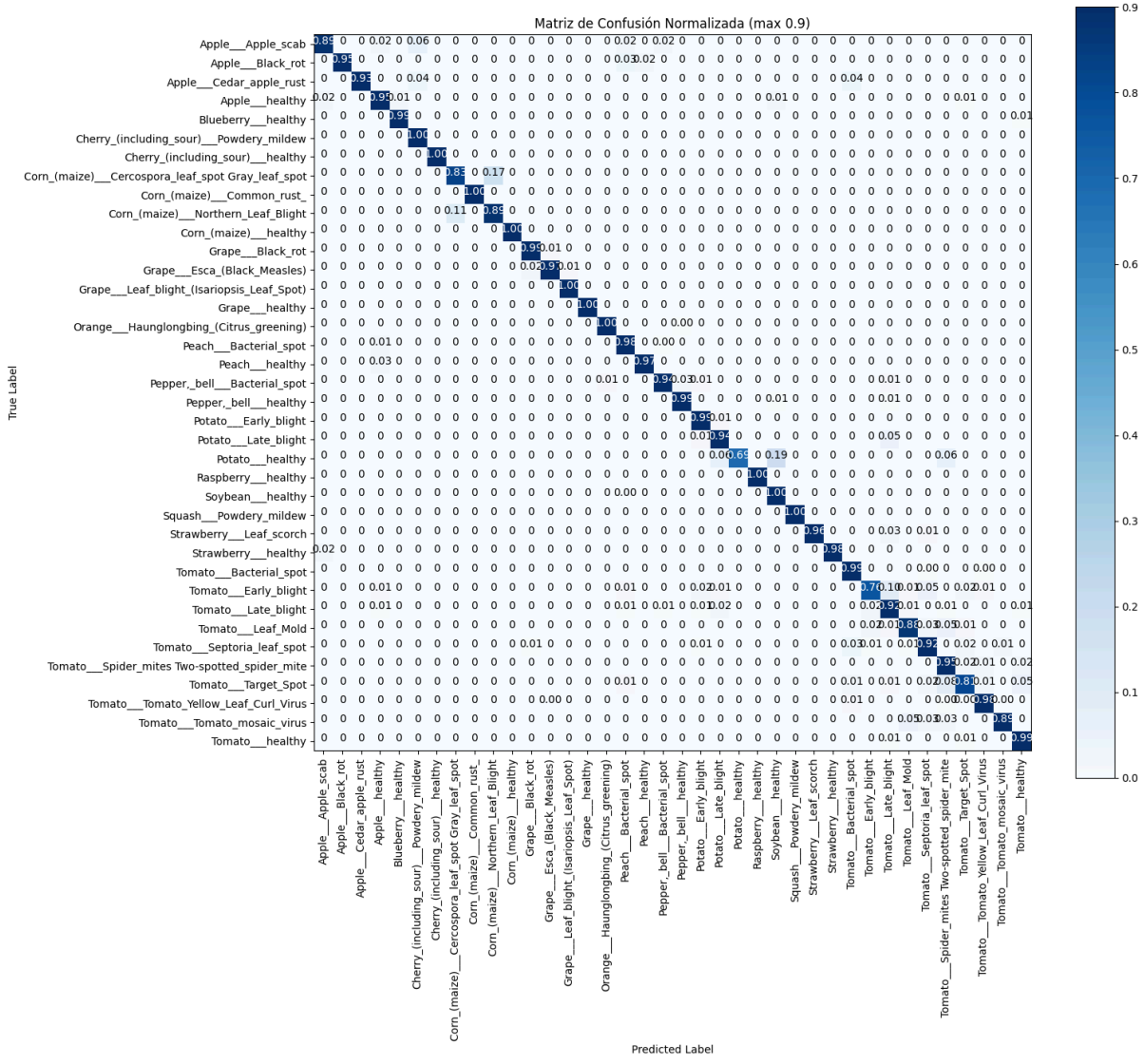


2. Evaluación en el conjunto de Test

El modelo alcanzó un rendimiento global muy destacado para ser un *baseline* sin ajuste fino (*fine-tuning*).

- Exactitud Global (Accuracy): El modelo obtuvo una exactitud del 96% sobre el conjunto de test (5459 imágenes).
- Métricas por clase: El reporte de clasificación muestra que la mayoría de las clases tienen una precisión y *recall* superiores al 95%, lo cual valida la elección de MobileNetV2 como un extractor de características robusto para este dominio.

Para poder ver los casos en donde se presentaron confusiones realizamos una matriz de confusión.



Para poder identificar las clases que tuvieron mayores problemas en su clasificación presentamos la siguiente tabla con los casos de menor F1-scores.

Los casos dificiles para el modelo fueron de tres tipos

1) Entre dos especies sanas: *Potato___healthy* y *Soybean___healthy.*
2) Entre dos especies con la misma enfermedad: *Potato___Late_blight* y *Tomato___Late_blight*

3) Entre dos enfermedades en la misma especie:
   Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot y
   Corn_(maize)___Northern_Leaf_Blight

De estos errores el más grave sería el tercero porque un diagnóstico equivocado puede resultar en un tratamiento incorrecto y a la consecuente propagación de la enfermedad.

Para poder identificar las clases que tuvieron mayores problemas en su clasificación presentamos la siguiente tabla con los casos de menor F1-scores.

Clase	**Precisión**	**Recall**	**F1-Score**	**Support**
Corn - Cercospora / Gray leaf spot	0.80	0.83	0.81	52
Potato - healthy	1.00	0.69	0.81	16
Tomato - Early blight	0.92	0.76	0.83	100
Tomato - Target Spot	0.90	0.81	0.85	141
Apple - Apple scab	0.92	0.89	0.90	63
Corn - Northern Leaf Blight	0.91	0.89	0.90	99
Tomato - Late blight	0.89	0.92	0.90	192
Tomato - Leaf Mold	0.93	0.88	0.90	96

Tomato - Mosaic virus	0.92	0.89	0.91	38
Tomato - Spider mites / Two-spotted mite	0.88	0.95	0.91	169

Trabajo futuro

Con el fin de mejorar los resultados, se analizará la necesidad de:

- Aplicar data augmentation.
- Utilizar alguno de los otros modelos disponibles en Pytorch (por ejemplo, mobilenet_v3_small)
- Utilización de otros optimizadores.
- Modificar la capa de clasificación y analizar otras alternativas.
- Agregar como input a la red las features que analizamos en el EDA que podrían ser útiles.

Nos gustaría también encontrar una red que reduzca el tiempo de ejecución habilitando el uso del modelo en aplicaciones móviles. Entre las estrategias de reducción de tamaño podríamos aplicar pruning para disminuir el número de pesos y cuantización para minimizar el uso de memoria y tiempo de cómputo.