Informe Ejecutivo –Clasificación de Enfermedades en Tomates con Deep Learning: Un Modelo CNN para la Agricultura

1st Ana María Vega Angarita *Estudiante* *Departamento de ingeniería de sistemas* *Universidad de Antioquia* *Medellín, Colombia* *ana.vega@udea.edu.co*	*2rd Maritza Tabarez Cárdenas* *Estudiante* *Departamento de ingeniería de sistemas* *Universidad de Antioquia* *Medellín, Colombia* *maritza.tabarezc@udea.edu.co*	*3th Johan Sebastian Henao Cañas* *Estudiante* *Departamento de ingeniería de sistemas* *Universidad de Antioquia* *Medellín, Colombia* *johan.henao1@udea.edu.co*

Acceso y Ejecución de los Notebooks

Para asegurar la correcta ejecución de todos los notebooks del proyecto, es necesario descargar el conjunto de datos como se indica en cada notebook.

1. Estructura de los Notebooks Entregados

El proyecto está organizado en estos notebooks principales:

- **01 - Exploración de datos.ipynb**: Se centra en la descarga, descompresión y análisis exploratorio del dataset de hojas de tomate. Incluye la visualización de clases, distribución de muestras.
- **02 - Preprocesamiento:** Se realiza la carga del dataset, análisis exploratorio, conteo de clases, visualización de imágenes y aplicación de técnicas de aumento de datos.
- **03 - ResNet-50:** Se entrena un modelo basado en ResNet50 con las capas convolucionales congeladas, luego se descongelan las últimas capas de ResNet50 y se entrena nuevamente el modelo para refinar su rendimiento, ajustando características más profundas a los datos específicos.
- **04 - Arquitectura MobileNetV3.ipynb**: Implementa la arquitectura *MobileNetV3* con transferencia de aprendizaje. Incluye preprocesamiento, construcción del modelo, entrenamiento, evaluación, visualización de resultados y análisis de métricas.
- **05 - Arquitectura InceptionV3.ipynb:** implementa la arquitectura InceptionV3, un modelo de redes neuronales convolucionales desarrollado por Google Research (*Google Brain*) y presentado en 2015 en el artículo *"Rethinking the Inception Architecture for Computer Vision*". *InceptionV3* es una evolución eficiente y profunda de la serie Inception, entrenada originalmente con el conjunto de datos *ImageNet*, y es ampliamente utilizada en tareas de clasificación de imágenes por su balance entre precisión y eficiencia computacional.
- **06 - Arquitectura VGG16.ipynb:** Se utilizó *VGG16*, una red convolucional profunda propuesta por *Simonyan* y *Zisserman* en 2014, caracterizada por su arquitectura

homogénea de 16 capas y exclusivamente filtros 3x3, con *max pooling* intermedio y capas densas finales. Este modelo es especialmente reconocido por su capacidad de extraer patrones visuales detallados, algo crítico para clasificar enfermedades con diferencias sutiles entre clases.

2. Descripción de la Solución - Modelos Entrenados

2.1 Arquitectura ResNet-50:

- Se utilizó *ResNet50* pre entrenado en *ImageNet* como base.
- Inicialmente se congelaron sus capas y se añadió una cabeza personalizada con *GlobalAveragePooling2D, Dense(1024) y Dense(num_classes)*.
- En el *fine-tuning*, se descongelaron las últimas 30 capas y se entrenó con una tasa de aprendizaje más baja para afinar los pesos.

Preprocesamiento:

- Las imágenes se redimensionan a 224x224 y se normalizaron (rescale=1./255).
- Se aplicó aumento de datos (rotación, zoom, desplazamientos y flip horizontal) para mejorar la generalización del modelo.

Datos:

- Dataset de Kaggle con 10 clases (9 enfermedades + 1 clase saludable).
- División: 10,000 imágenes para entrenamiento y 1,000 para validación.

2.2 Arquitectura MobileNetV3:

- Se utilizó *MobileNetV3-Small* preentrenada en *ImageNet*.
- Se aplicó *fine-tuning* descongelando las últimas 30 capas.
- El modelo se compuso de una capa de entrada, bloque de aumento de datos, *MobileNetV3* base congelado parcialmente, *GlobalAveragePooling2D* y capa densa final con activación *softmax*.

Preprocesamiento:

- Imágenes redimensionadas a 224x224 px.
- Normalización con mobilenet_v3.preprocess_input.
- Aumento de datos mediante rotación, zoom, contraste, brillo y volteo aleatorio.

Datos:

- *Dataset* de *Kaggle* con 10 clases (9 enfermedades + 1 clase saludable).
- División: 10,000 imágenes para entrenamiento y 1,000 para validación.

2.3 Arquitectura InceptionV3:

Se utilizó la arquitectura *InceptionV3* preentrenada en *ImageNet*, desarrollada por *Google Research.*

- Se aplicó *fine-tuning*, descongelando las últimas capas del modelo base.
- Se construyó el modelo con las siguientes capas:
  - Entrada (224x224x3)
  - Bloque de aumento de datos (rotación, zoom, brillo, etc.)
  - *InceptionV3* base con pesos de *ImageNet* y capas superiores entrenables
  - *GlobalAveragePooling2D*
  - Capa *Dropout(0.2)*
  - Capa densa final con activación *softmax* (10 clases)

Preprocesamiento:

- Redimensionamiento a 224x224 px
- Preprocesamiento con *inception_v3.preprocess_input()*
- Aumento de datos con transformaciones aleatorias

Datos:

- Dataset de *Kaggle* con 10 clases (9 enfermedades + 1 clase saludable)
  División: 10,000 imágenes para entrenamiento y 1,000 para validación

2.4 Arquitectura VGG16:

Preprocesamiento:

- Redimensionamiento de imágenes a 224×224 píxeles.
- Normalización mediante *tf.keras.applications.vgg16.preprocess_input()*.
- Aumento de datos intensivo (rotaciones, zoom, contraste y volteo aleatorio) para mitigar overfitting.

Modelo base:

- VGG16 preentrenado en ImageNet con include_top=False, es decir, excluyendo las capas densas originales.
- Inicialmente todas sus capas fueron congeladas.

Cabeza personalizada:

- *GlobalAveragePooling2D* para reducir la dimensionalidad de los mapas de activación.
- Capa *Dropout*(0.3) como regularización.
- Capa *Dense*(num_classes, activation="softmax") como clasificador final.

3. Iteraciones Realizadas

3.1 ResNet-50

Entrenamiento ResNet-50

- Se congelaron todas las capas de *ResNet-50.*
- Se entrenó el clasificador superior durante 10 épocas con *EarlyStopping*.
  Se logró una mejora moderada en la precisión y pérdida.

Fine-tuning

- Se descongelaron las últimas 30 capas de *ResNet-50.*
- Se recompiló el modelo con una tasa de aprendizaje más baja (1e-5).
- Se entrenó durante 5 épocas adicionales.
- Se observó una ligera mejora en la precisión y una reducción constante de la pérdida en validación.

3.2 MobileNetV3

- **Primera iteración:** Entrenamiento con *MobileNetV3* base congelado; resultados aceptables pero con overfitting temprano.
- **Segunda iteración:** *Fine-tuning* parcial con últimas 30 capas activas, optimización con Adam, early stopping y ajuste de tasa de aprendizaje.
- **Tercera iteración:** Ajustes al bloque de aumento de datos y regularización con Dropout(0.2).

3.3 InceptionV3

- **Primera iteración:** Entrenamiento con la arquitectura base de InceptionV3 completamente congelada y solo las capas finales entrenables. El modelo alcanzó un rendimiento decente, pero presentaba estancamiento temprano en la mejora de la precisión de validación.
- **Segunda iteración:** Se aplicó *fine-tuning* activando las últimas 30 capas del modelo base. Se utilizó el optimizador Adam con una tasa de aprendizaje inicial de 1e-3, y callbacks como *EarlyStopping* y *ModelCheckpoint* para evitar sobreentrenamiento. Se observaron mejoras en las métricas de validación.
- **Tercera iteración:** Se mantuvo el *fine-tuning* y se potenció el bloque de aumento de datos (flip, rotación, zoom, contraste, brillo), junto con regularización por *Dropout(0.2)*. El entrenamiento fue más estable, logrando una precisión de validación del 88 %.

3.4 VGG16

Entrenamiento inicial:

- Todas las capas convolucionales congeladas.
- Solo se entrenaron las capas densas superiores.
- El modelo alcanzó rápidamente una alta precisión en validación, evidenciando la utilidad del conocimiento preentrenado.

Fine-tuning:

- Se descongelaron las últimas 10 capas convolucionales de VGG16.
- Se recompiló con una tasa de aprendizaje reducida (1e-4) para permitir la adaptación gradual de los filtros.
- Este ajuste fino permitió al modelo especializarse en los patrones propios de las hojas de tomate, mejorando la generalización.

4. Resultados Obtenidos

4.1 ResNet-50

- La precisión de entrenamiento se mantuvo estable (~0.35–0.36) durante el fine-tuning.
- La precisión de validación aumentó de ~0.30 a ~0.325, lo cual indica que el modelo logró generalizar mejor.
- La pérdida de validación disminuyó progresivamente hasta ~1.87, demostrando una mejora en el ajuste del modelo a los datos reales.
- No se detectaron señales evidentes de sobreajuste gracias al uso de EarlyStopping y a la tasa de aprendizaje controlada.

4.2 MobileNetV3

- **Precisión final de validación:** 95.3% (mejor época)
- **Matriz de confusión:** Alto desempeño en la mayoría de clases; mínima confusión entre enfermedades similares.
- **Métricas por clase:**
  - *F1-score promedio:* 0.95
  - *Mejores clases:*
    - Tomato___Tomato_mosaic_virus (1.00)
    - Tomato___Bacterial_spot (1.00)
    - Tomato___healthy (1.00)
  - *Clase con menor recall:* Tomato___Spider_mites (0.83)
- **Gráficas:**
  - Precisión y pérdida mostraron convergencia estable.
  - Visualizaciones demostraron predicciones correctas y errores clasificados.

4.3 InceptionV3

- **Precisión final de validación:** 88.2% (mejor época)
- **Matriz de confusión:** Logra una clasificación eficaz en la mayoría de las clases, aunque existen algunas diferencias en rendimiento entre ellas.
- **Métricas por clase:**
  - *Mejore clase: Tomato___Tomato_Yellow_Leaf_Curl_Virus* (0.95)
  - *Clase con menor recall:* Tomato___healthy(0.82)
- **Gráficas:**
  - Precisión y pérdida mostraron convergencia estable.
  - Visualizaciones demostraron predicciones correctas y errores clasificados.

4.4 VGG16

- **Gráficas de desempeño:**

  La precisión de entrenamiento y validación alcanzó valores superiores al 95% desde las primeras 5 épocas.
  La pérdida descendió rápidamente por debajo de 0.2, mostrando una convergencia estable.
  No se observó sobreajuste severo gracias al uso de aumento de datos y EarlyStopping.

- **Matriz de confusión:**

  El modelo logró identificar correctamente casi todas las clases, con recall superior al 93% en todas las categorías.
  Clases con menor recall: Tomato___Spider_mites (0.93) y Tomato___Septoria_leaf_spot (0.94), que presentan patrones visuales muy similares.
  La clase Tomato___healthy fue clasificada correctamente en el 100% de los casos.

- **Métricas por clase (F1-score):**

  Promedio macro: 0.97
  Mejor clase: Tomato___healthy (F1=1.00)
  Peor clase: Tomato___Spider_mites (F1=0.93)

- **Visualización de predicciones:**

  Las imágenes predichas correctamente mostraron hojas con características bien definidas.
  Los pocos errores se concentraron en confusiones entre enfermedades de manchas foliares.

- **Precisión final:**

  Validación: 97%
  Entrenamiento: 98%

5. Conclusiones:

ResNet50 demostró ser una base sólida para la clasificación de hojas de tomate gracias a su capacidad de aprender representaciones profundas. Inicialmente, al congelar sus capas, el modelo alcanzó un rendimiento aceptable. Tras aplicar *fine-tuning* a sus capas finales, se logró una mejora en la precisión y una reducción de la pérdida, confirmando que ajustar parte de la arquitectura preentrenada permite adaptar mejor el modelo al dominio específico del problema.

El modelo basado en *MobileNetV3* demostró ser altamente efectivo y eficiente para la clasificación de enfermedades en hojas de tomate, logrando una precisión del 95% en validación. La arquitectura ligera y los mecanismos de regularización permitieron una generalización adecuada. Esta solución puede ser integrada en sistemas agrícolas de bajo costo para monitoreo de cultivos.

InceptionV3 logró una clasificación robusta de las 10 clases de enfermedades en hojas de tomate. Aunque el modelo confunde algunas clases visualmente similares, como enfermedades con patrones foliares parecidos. A pesar de ello, no hay clases ignoradas, lo que demuestra que el modelo aprendió a distinguir todas las categorías con un rendimiento razonable.

El modelo *VGG16* demostró un rendimiento sobresaliente en la clasificación de enfermedades en hojas de tomate. Su arquitectura profunda, combinada con la transferencia de conocimiento de *ImageNet* y el *fine-tuning* controlado, permitió alcanzar una precisión de validación del 97%. La estabilidad de la convergencia, la robustez frente al *overfitting* y la claridad en la diferenciación de clases hacen de esta solución una alternativa de alta calidad para su aplicación en sistemas agrícolas asistidos por inteligencia artificial. Este enfoque puede ser integrado en herramientas de diagnóstico automatizado que requieren modelos con gran capacidad de generalización y precisión, especialmente en entornos con clases de aspecto visual altamente similar.

En términos generales, el modelo que alcanzó el mejor rendimiento global fue *VGG16*, con una precisión de validación del 97%, un F1-score promedio de 0.97, y una matriz de confusión que muestra un reconocimiento prácticamente perfecto en la mayoría de las clases. *MobileNetV3* se ubicó muy cerca, con una precisión final de validación del 95.3% y un excelente balance entre eficiencia computacional y precisión, destacándose especialmente por su capacidad de generalizar sin requerir un alto consumo de memoria, por lo que sería la opción más adecuada en sistemas embebidos o móviles. *InceptionV3* logró una precisión de 88.2%, mostrando buen desempeño pero menor capacidad para diferenciar clases muy similares. Finalmente, *ResNet-50*, a pesar de su

arquitectura profunda y su efectividad comprobada en otros dominios, fue el que obtuvo los resultados más discretos en este problema específico (precisión cercana al 32%), probablemente debido a la necesidad de más entrenamiento o un ajuste más fino de hiperparámetros. En conclusión, si la prioridad es la máxima precisión en clasificación de enfermedades de hojas de tomate, *VGG16* se perfila como la solución más robusta, mientras que *MobileNetV3* ofrece el mejor compromiso entre desempeño y eficiencia.

Referencias

[1] https://rramosp.github.io/2021.deeplearning/content/M04.html
[2] **https://www.kaggle.com/datasets/kaustubhb999/tomatoleaf/**