Clasificación de Enfermedades del Arroz usando Transfer Learning (VGG16)

Este informe describe la estructura y los resultados del análisis y modelado implementado para la clasificación de enfermedades en imágenes de hojas de arroz.

---

1. Estructura del Notebook

El notebook sigue la siguiente estructura para abordar el problema de clasificación de imágenes:

- **Descarga y Preparación de Datos:**
  Se inicia con la descarga del dataset de Kaggle para lo cual basta ejecutar la segunda celda e ingresar las credenciales obtenidas en el ***kaggle.json***. Luego, se filtra el dataset original para incluir únicamente las cinco clases de interés: bacterial leaf blight, brown spot, hispa, tungro y normal.
  Las imágenes de estas clases se copian a un nuevo directorio. Posteriormente, se dividen aleatoriamente en conjuntos de entrenamiento (80%) y validación (20%), manteniendo la estructura de carpetas por clase para su uso con generadores de imágenes. Se verifica el conteo de imágenes por clase en cada conjunto.

- **Definición y Construcción del Modelo:**
  Se define un modelo de red neuronal convolucional utilizando Transfer Learning. Se emplea la arquitectura VGG16 preentrenada con pesos de ImageNet, excluyendo la capa densa superior. Luego, se añaden capas personalizadas para la clasificación de las cinco clases específicas.

- **Entrenamiento del Modelo:**
  El modelo se compila con un optimizador y función de pérdida apropiados para clasificación multiclase. Se entrena usando los generadores previamente creados, iterando durante varias épocas. Se utiliza un callback que guarda automáticamente el modelo con la mejor precisión en el conjunto de validación.

- **Evaluación del Modelo:**
  Una vez completado el entrenamiento, el modelo se evalúa en el conjunto de validación mediante precisión global, reporte de clasificación por clase (precisión, recall, F1-score) y matriz de confusión. Se visualizan las curvas de precisión y pérdida para analizar el aprendizaje.

- **Inferencia (Ejemplo):**
  Se proporciona un ejemplo de cómo cargar el mejor modelo guardado y realizar una predicción sobre una imagen individual.

---

2. Arquitectura del Modelo: VGG16 con Transfer Learning

La solución de modelado se basa en la arquitectura VGG16, una red convolucional profunda pre-entrenada con el dataset ImageNet.

- **Base del Modelo:**
  Se cargó la arquitectura `VGG16` desde la API de Keras con `weights=imagenet` y `include_top=False`, lo que permite aprovechar sus pesos sin incluir las capas densas superiores (específicas para las clases de ImageNet).

- **Congelación de Capas Base:**
  Las capas convolucionales y de pooling fueron congeladas (`base_model.trainable = False`), es decir, sus pesos no se actualizaron durante el entrenamiento. Esto permite conservar la capacidad de extracción de características de VGG16.

- **Capas Adicionales:**
  Sobre la base congelada, se añadieron capas densas personalizadas:

  - `Flatten` para aplanar la salida convolucional,
  - `BatchNormalization` para regularización,
  - Capas `Dense` con activación `relu`,
  - `Dropout` para prevenir sobreajuste,
  - y una capa `Dense` final con activación `softmax` para predecir las cinco clases.

---

3. Preprocesamiento de Imágenes

El preprocesamiento fue esencial para adaptar los datos al modelo VGG16:

- **Normalización:**
  Se utilizó `ImageDataGenerator` con `rescale=1./255` para escalar los valores de píxel de [0, 255] a [0, 1], lo cual mejora el entrenamiento.

- **Generadores de Datos:**
  Se crearon `train_generator` y `val_generator`, los cuales leen imágenes desde las carpetas respectivas, redimensionan a `(224, 224)`, y generan lotes con etiquetas codificadas en one-hot (`class_mode=categorical`).
  El generador de entrenamiento usa `shuffle=True` para aleatorizar las imágenes

en cada época.

---

4. Iteraciones de Entrenamiento

El modelo se entrenó durante **15 épocas**:

- **Épocas:**
  Se realizaron 15 iteraciones completas (`epochs=15`) sobre el conjunto de entrenamiento.

- **Monitoreo y Guardado:**
  Se empleó un callback ModelCheckpoint que monitorea la val_accuracy y guarda el modelo en /content/mejor_modelo_vgg16_gpu.h5 solo si hay mejora (save_best_only=True, mode='max').

- **Curvas de Entrenamiento:**
  Las curvas mostraron un rápido aumento en la precisión del entrenamiento (cercana al 100%) y mejora significativa en validación, aunque con fluctuaciones.
  La pérdida en entrenamiento disminuyó constantemente; en validación también bajó, aunque con valores algo más altos.
  Se observó un leve sobreajuste hacia el final, mitigado por el `ModelCheckpoint`.

---

5. Resultados Clave

La evaluación final del modelo arrojó los siguientes resultados:

- **Precisión Global en Validación:**
  El modelo alcanzó una precisión del **93.47%** sobre el conjunto de validación.

- **Reporte de Clasificación por Clase:**

  - **bacterial_leaf_blight:** Precisión 0.98, Recall 0.82, F1-score 0.89
  - **brown_spot:** Precisión 0.94, Recall 0.93, F1-score 0.94
  - **hispa:** Precisión 0.94, Recall 0.96, F1-score 0.95
  - **normal:** Precisión 0.95, Recall 0.97, F1-score 0.96
  - **tungro:** Precisión 0.95, Recall 0.94, F1-score 0.95

- Se observa un buen rendimiento en general. La clase bacterial_leaf_blight tiene alta precisión pero un recall menor, lo que sugiere que el modelo identifica bien los casos que clasifica como tal, pero omite algunos positivos reales.

- **Matriz de Confusión:**

  La matriz muestra valores diagonales altos, reflejando una clasificación correcta en la mayoría de los casos. Los errores fuera de la diagonal indican confusiones entre clases específicas, útiles para análisis futuro.

---

6. Conclusiones

El uso de Transfer Learning con VGG16, junto con un preprocesamiento adecuado, ha permitido desarrollar un modelo con un rendimiento sólido para clasificar cinco enfermedades del arroz (y hojas sanas), lo que lo convierte en una herramienta prometedora para aplicaciones en el sector agrícola.