Instituto tecnológico de Culiacán

Ingeniería en tecnologías de la información y comunicaciones

Tópicos de Inteligencia Artificial

Unidad 4

Tarea 2

“Desarrollo de modelo de visión artificial para clasificar plantas”

Jesús Alberto Barraza Castro

Jesús Guadalupe Wong Camacho

30/05/2025

Introducción

Este documento detalla el proceso de creación de un modelo de clasificación de plantas utilizando Redes Neuronales Convolucionales (CNNs). El objetivo principal es construir un modelo capaz de identificar diferentes especies de plantas a partir de imágenes, empleando las capacidades de las CNNs para el reconocimiento de patrones visuales.

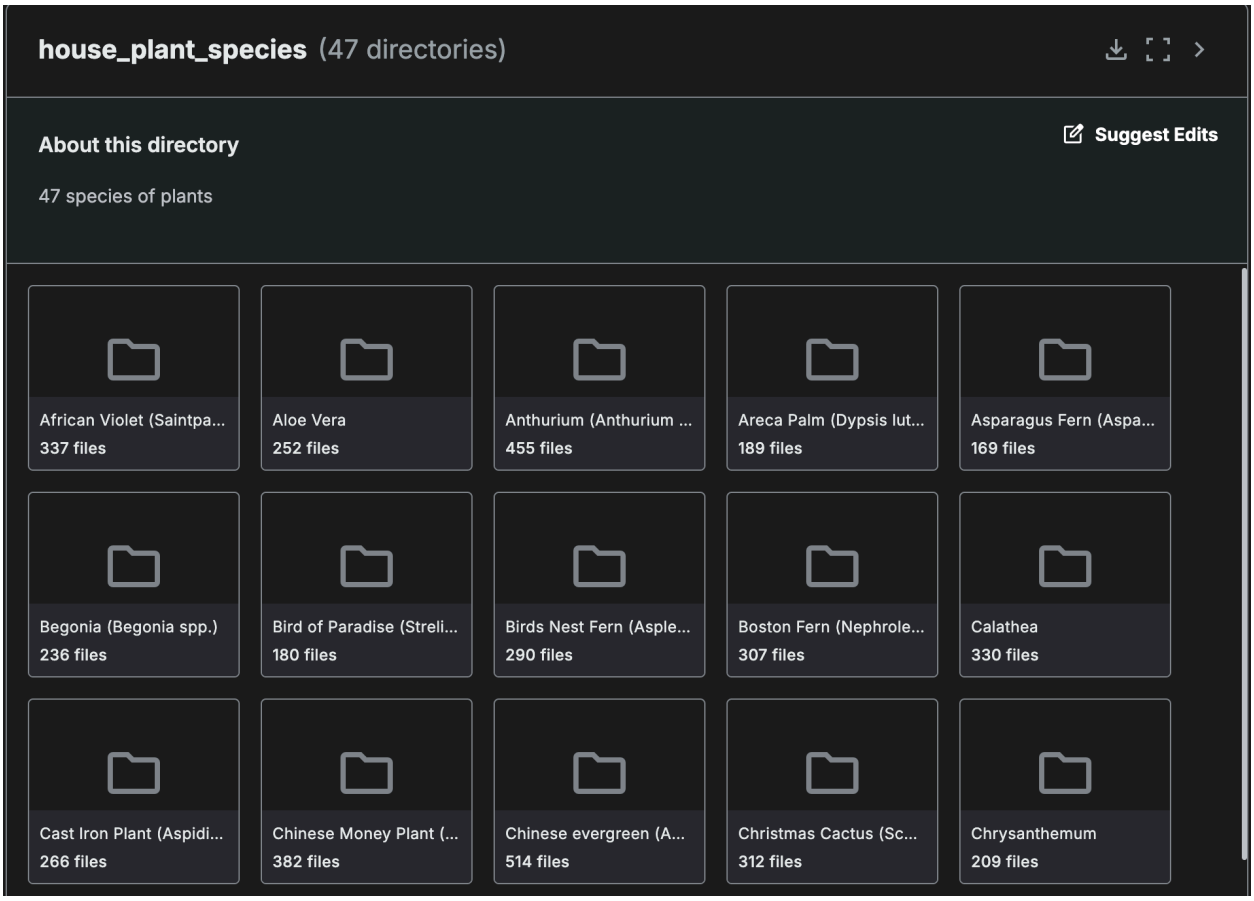
Configuración del Entorno y Librerías

Para el desarrollo del modelo se utilizaron las siguientes librerías principales en Python:

- **pandas**: Para manipulación y análisis de datos.
- **numpy**: Para operaciones numéricas eficientes.
- **matplotlib.pyplot y seaborn**: Para visualización de datos y resultados.
- **tensorflow y tensorflow.keras:** Para la construcción, entrenamiento y evaluación de la red neuronal.
- **sklearn.metrics:** Para métricas de evaluación del modelo, como el classification_report y confusion_matrix.
- **os:** Para interactuar con el sistema de archivos (creación de directorios, manejo de rutas).
- **warnings:** Para controlar la visualización de advertencias.

Fuente de datos:

Las imágenes utilizadas para entrenar y evaluar este modelo fueron obtenidas del dataset "House Plant Species" disponible en Kaggle, en la siguiente URL: https://www.kaggle.com/datasets/kacpergregorowicz/house-plant-species/data.
Este dataset contiene imágenes de diversas especies de plantas, adecuadamente etiquetadas para tareas de clasificación.



Preparación de los datos:

El primer paso en la preparación de los datos fue la organización de la estructura de directorios. Se crearon tres directorios principales:

- Train: Contiene las imágenes destinadas al entrenamiento del modelo.
- Test: Contiene imágenes utilizadas para ajustar los hiperparámetros del modelo y monitorear el rendimiento durante el entrenamiento, previniendo el sobreajuste.
- Prediction: Contiene imágenes completamente nuevas, no vistas por el modelo durante el entrenamiento o la validación, utilizadas para la evaluación final del rendimiento generalizado del modelo.

Se destinó un 60% de los datos para entrenamiento, 30% para la validación y el 10% restante se guardaron para el directorio de Prediction para realizar las pruebas finales con imagenes que no ha visto el modelo.

Dentro de cada uno de estos directorios, se encuentran varios directorios mas, cada uno representando una clase de planta específica y dentro de los cuales se encuentran las imágenes.

Pre-procesamiento de los datos:

Las imágenes fueron preprocesadas antes de ser alimentadas al modelo:

- **Redimensionamiento:** Todas las imágenes se redimensionaron a un tamaño uniforme de (128, 128) píxeles. Esto asegura que todas las entradas al modelo tengan la misma dimensión, lo cual es un requisito para las capas convolucionales.
- **Normalización:** Los valores de los píxeles, que originalmente suelen estar en el rango de 0 a 255, fueron reescalados a un rango más pequeño (comúnmente entre 0 y 1) dividiéndolos por 255.

Definición de la arquitectura de la red neuronal

La arquitectura seleccionada fue la de la Red Neuronal Convolucional.
Se construye utilizando el modelo Sequential de Keras, lo que permite apilar capas de manera lineal. La arquitectura diseñada busca extraer características jerárquicas de las imágenes y clasificarlas.

Las capas utilizadas fueron:

- **Capa de Entrada (Input):** Define la forma de las imágenes de entrada al modelo.
- **Bloques Convolucionales** (Convolución + Activación + Pooling): El modelo utiliza una serie de bloques convolucionales para aprender patrones de las imágenes. Cada bloque consta de una capa Conv2D seguida de una función de activación y una capa MaxPooling2D.
- **Capa de Aplanamiento (Flatten):** Transforma la salida tridimensional de las capas convolucionales en un vector unidimensional, lo cual es necesario para conectar las capas convolucionales con las capas densas.
- Capas densas: Estas capas son las responsables de la clasificación final, utilizando las características extraídas por las capas convolucionales.

La arquitectura sigue una secuencia de capas convolucionales y de pooling para la extracción de características, seguida de capas densas y de dropout para la clasificación final y la regularización.

Entrenamiento del modelo

El modelo se entrenó utilizando los datos de entrenamiento y se valida con el conjunto de test. Se utiliza un EarlyStopping para monitorear la precisión de la validación y detener el entrenamiento si no mejora después de cierto número de epochs y asi podemos evitar overfitting.

Pruebas del modelo final

Prueba de precisión global:

```
model.evaluate(test_ds)
✓ 17.4s
222/273 ━━━━━━━━━━━━━━━━━━━━ 3s 62ms/step - accuracy: 0.6761 - loss: 1.1631
2025-05-30 18:17:08.071286: W tensorflow/core/lib/png/png_io.cc:89] PNG warning: iCCP: known incorrect sRGB profile
240/273 ━━━━━━━━━━━━━━━━━━━━ 2s 62ms/step - accuracy: 0.6759 - loss: 1.1641
2025-05-30 18:17:09.097418: W tensorflow/core/lib/png/png_io.cc:89] PNG warning: iCCP: known incorrect sRGB profile
273/273 ━━━━━━━━━━━━━━━━━━━━ 17s 62ms/step - accuracy: 0.6758 - loss: 1.1651

[1.175206184387207, 0.6753662824630737]
```

Al evaluar el modelo con los datos de prueba, se obtuvo una precisión de aproximadamente 67.5%. Esto significa que el modelo acierta en casi 7 de cada 10 predicciones.