

INSTITUTO TECNOLÓGICO NACIONAL DE MÉXICO CAMPUS CULIACÁN

TOPICOS DE IA



Alumno:

Portillo Zuñiga Steve Javier

Franco Flores Luis Fernando

Unidad 4 Tarea 2

Hora: 10:00 – 11:00

Maestro: Zuriel Dathan Mora Felix

Indice

Construcción del Dataset.....	3
Arquitectura del Modelo CNN	3
Implementación del Modelo.....	4
Evaluación y Pruebas.....	6

Construcción del Dataset

Se construyó un dataset personalizado con 55 clases distintas de plantas, asegurando variedad en forma, color y estructura de hojas o flores. La fuente de datos consistió en imágenes recopiladas de internet

Limpieza:

- Se validaron imágenes con un script para verificar formato, resolución mínima (224x224) y canales RGB.
- Se eliminaron imágenes corruptas o mal etiquetadas.

Transformación:

- Todas las imágenes fueron redimensionadas a 224x224 píxeles.
- Se normalizaron los valores de píxeles al rango [0, 1] para facilitar el aprendizaje del modelo..

```
PS C:\Users\STIV\Desktop\Bot> & C:\Users\STIV\AppData\Local\Microsoft\WindowsApps\python3.12.exe C:/Users/STIV/Desktop/Bot/ml_aplicacion/procesar_dataset_plantas.py
[african_violet] entrenamiento: 100%
[african_violet] validación: 100%
[aglaonema] entrenamiento: 100%
[aglaonema] validación: 100%
[aloe_vera] entrenamiento: 100%
[aloe_vera] validación: 100%
[anthurium] entrenamiento: 100%
[anthurium] validación: 100%
[areca_palm] entrenamiento: 100%
[areca_palm] validación: 100%
[bamboo_plant] entrenamiento: 100%
[bamboo_plant] validación: 100%
[basil_plant] entrenamiento: 100%
[basil_plant] validación: 100%
[begonia] entrenamiento: 100%
[begonia] validación: 100%
[boston_fern] entrenamiento: 100%
[boston_fern] validación: 100%
[cactus] entrenamiento: 100%
[cactus] validación: 100%
[caladium] entrenamiento: 100%
[caladium] validación: 100%
[calathea] entrenamiento: 100%
[calathea] validación: 100%
[camellia] entrenamiento: 100%
[camellia] validación: 100%
[chinese_evergreen] entrenamiento: 100%
[chinese_evergreen] validación: 100%
[chrysanthemum] entrenamiento: 100%
[chrysanthemum] validación: 100%
[coleus] entrenamiento: 100%
68/68 [00:02:00:00, 27.88it/s]
18/18 [00:00:00:00, 28.74it/s]
66/66 [00:03:00:00, 21.90it/s]
17/17 [00:00:00:00, 19.74it/s]
72/72 [00:02:00:00, 33.68it/s]
19/19 [00:00:00:00, 63.11it/s]
60/60 [00:02:00:00, 28.64it/s]
16/16 [00:00:00:00, 24.75it/s]
75/75 [00:02:00:00, 33.08it/s]
19/19 [00:00:00:00, 45.12it/s]
72/72 [00:02:00:00, 29.78it/s]
18/18 [00:00:00:00, 20.22it/s]
65/65 [00:02:00:00, 23.94it/s]
17/17 [00:00:00:00, 37.11it/s]
71/71 [00:03:00:00, 18.68it/s]
19/19 [00:00:00:00, 32.53it/s]
72/72 [00:02:00:00, 30.97it/s]
19/19 [00:00:00:00, 35.44it/s]
72/72 [00:02:00:00, 24.84it/s]
18/18 [00:00:00:00, 24.74it/s]
70/70 [00:02:00:00, 19.61it/s]
20/20 [00:00:00:00, 32.89it/s]
76/76 [00:03:00:00, 23.85it/s]
20/20 [00:00:00:00, 25.28it/s]
68/68 [00:02:00:00, 25.44it/s]
18/18 [00:00:00:00, 26.90it/s]
68/68 [00:02:00:00, 26.96it/s]
17/17 [00:00:00:00, 62.03it/s]
60/60 [00:02:00:00, 25.01it/s]
16/16 [00:00:00:00, 56.93it/s]
68/68 [00:02:00:00, 26.06it/s]
```

Arquitectura del Modelo CNN

Se utilizó una red neuronal convolucional clásica (CNN) definida de la siguiente forma:

- Entrada: Imagen RGB (224x224x3)
- Capas:
 - Conv2D (32 filtros, 3x3, ReLU) + MaxPooling2D
 - Conv2D (64 filtros, 3x3, ReLU) + MaxPooling2D
 - Conv2D (128 filtros, 3x3, ReLU) + MaxPooling2D
 - Flatten + Dense(256, ReLU) + Dropout(0.5)

- Dense de salida con softmax y 55 clases

Esta arquitectura balancea precisión con velocidad de entrenamiento.

Implementación del Modelo

Tecnologías usadas:

- TensorFlow 2.18
- Keras
- OpenCV
- scikit-learn, NumPy

Entrenamiento y Validación:

- Dataset: 5482 imágenes de entrenamiento, 469 imágenes de validación
- Épocas: 10
- Optimización: Adam
- Pérdida: Categorical Crossentropy
- Métrica: Accuracy

Código Base:

```
model = Sequential([
    Conv2D(32, (3,3), activation='relu', input_shape=(224,224,3)),
    MaxPooling2D(2,2),
    Conv2D(64, (3,3), activation='relu'),
    MaxPooling2D(2,2),
    Conv2D(128, (3,3), activation='relu'),
    MaxPooling2D(2,2),
    Flatten(),
    Dense(256, activation='relu'),
    Dropout(0.5),
    Dense(55, activation='softmax')
])
```

```
model.compile(optimizer='adam', loss='categorical_crossentropy',  
metrics=['accuracy'])  
  
model.fit(train_gen, validation_data=val_gen, epochs=10)  
  
model.save("modelo_plantas.h5")
```

```
Epoch 1/10  
469/469 ————— 421s 894ms/step - accuracy: 0.0724 - loss: 3.7526 - val_accuracy: 0.2045 - val_loss: 2.9401  
Epoch 2/10  
469/469 ————— 337s 719ms/step - accuracy: 0.2312 - loss: 2.7960 - val_accuracy: 0.2696 - val_loss: 2.7219  
Epoch 3/10  
469/469 ————— 330s 703ms/step - accuracy: 0.4242 - loss: 2.0400 - val_accuracy: 0.3275 - val_loss: 2.6998  
Epoch 4/10  
469/469 ————— 341s 727ms/step - accuracy: 0.6550 - loss: 1.1739 - val_accuracy: 0.3182 - val_loss: 3.1921  
Epoch 5/10  
469/469 ————— 395s 843ms/step - accuracy: 0.8139 - loss: 0.6145 - val_accuracy: 0.3326 - val_loss: 3.9275  
Epoch 6/10  
469/469 ————— 365s 778ms/step - accuracy: 0.8848 - loss: 0.3882 - val_accuracy: 0.3140 - val_loss: 4.3891  
Epoch 7/10  
469/469 ————— 383s 816ms/step - accuracy: 0.9165 - loss: 0.2746 - val_accuracy: 0.3068 - val_loss: 4.9450  
Epoch 8/10  
355/469 ————— 1:28 775ms/step - accuracy: 0.9436 - loss: 0.1913
```

Evaluación y Pruebas

Métricas utilizadas:

- Accuracy (exactitud global)
- Precision y Recall por clase

Pruebas con Cámara Web:

Se implementó un sistema en tiempo real que:

- Captura imagen con cv2.VideoCapture
- Preprocesa el frame
- Realiza inferencia con el modelo entrenado
- Muestra la etiqueta predicha sobre el video

```
cap = cv2.VideoCapture(0)
```

```
while True:
```

```
    ret, frame = cap.read()
```

```
    img = cv2.resize(frame, (224, 224))
```

```
    img_array = np.expand_dims(img / 255.0, axis=0)
```

```
    prediction = model.predict(img_array)
```

```
    label = labels[np.argmax(prediction)]
```

```
    cv2.putText(frame, f"Prediccion: {label}", (20, 40),  
cv2.FONT_HERSHEY_SIMPLEX, 1, (0,255,0), 2)
```

```
    cv2.imshow("Clasificador de Plantas", frame)
```

```
    if cv2.waitKey(1) & 0xFF == ord('q'):
```

```
        break
```



