Modelo de clasificación de algunas Aves del Tolima

Daniel Felipe Medina Moreno

ID 774065

Ing. Oscar Diaz

Universidad Cooperativa de Colombia sede Ibagué

Ingeniería de sistemas

2024

Tabla de contenido

Introducción

El propósito principal de este proyecto es crear un modelo avanzado para clasificar aves utilizando técnicas modernas de aprendizaje profundo. Para ello, se adoptó la metodología CRISP-DM (Cross-Industry Standard Process for Data Mining), un enfoque estructurado que guía cada etapa del desarrollo, desde la comprensión inicial del problema hasta la implementación de una solución efectiva.

En el entrenamiento del modelo, las imágenes fueron sometidas a un proceso de preprocesamiento que elimina el fondo y enfatiza exclusivamente las características del ave. Este procedimiento busca reducir el ruido en los datos y mejorar la precisión del modelo al centrarse en los aspectos más relevantes de las imágenes.

El desarrollo del modelo se apoyó en técnicas de Transfer Learning, utilizando arquitecturas preentrenadas como ResNet50 y VGG16. Estas redes, diseñadas para trabajar con grandes volúmenes de datos, se adaptaron específicamente al contexto de la clasificación de aves, aprovechando su conocimiento previo. Este enfoque no solo incrementó la precisión del modelo, sino que también optimizó el tiempo de inferencia, haciendo que la solución sea ideal para aplicaciones en tiempo real.

Comprensión del negocio

El proyecto busca desarrollar un modelo de clasificación de imágenes que permita identificar diferentes especies de aves de manera precisa y eficiente. Para lograr esto, no solo se enfoca en la construcción del modelo, sino también en maximizar su rendimiento mediante técnicas avanzadas como el preprocesamiento de imágenes, que elimina elementos distractores (como el fondo) y resalta las características únicas de las aves. Este enfoque estratégico reduce el ruido en los datos y asegura que el modelo se concentre en los aspectos más relevantes para la tarea de clasificación.

Comprensión de los datos

Las imágenes utilizadas en el entrenamiento son fundamentales para el éxito del modelo. El hecho de que las imágenes provengan de fuentes diversas, como Google y la aplicación eBird, garantiza una amplia variedad de contextos y condiciones, lo que puede mejorar la generalización del modelo. Sin embargo, también introduce desafíos relacionados con la calidad y homogeneidad de los datos, como posibles variaciones en iluminación, resolución y ángulos de captura.

Es crucial analizar a fondo la distribución de clases, ya que un desequilibrio en la representación de especies podría generar sesgos en el modelo, favoreciendo la identificación de especies más comunes y dificultando el reconocimiento de las menos frecuentes. Esto se refleja en los resultados de la matriz de confusión, donde es evidente que algunas clases están mejor representadas y reconocidas que otras.

Preprocesamiento de los datos

El pre procesamiento adicional, como la eliminación de fondos para destacar únicamente las aves, juega un papel crucial en mitigar estos desafíos al reducir el ruido en las imágenes. Sin embargo, se podrían considerar estrategias adicionales, como técnicas de aumento de datos (data augmentation) para equilibrar la representación de las clases menos comunes o ponderar las pérdidas durante el entrenamiento para dar mayor importancia a las especies subrepresentadas.

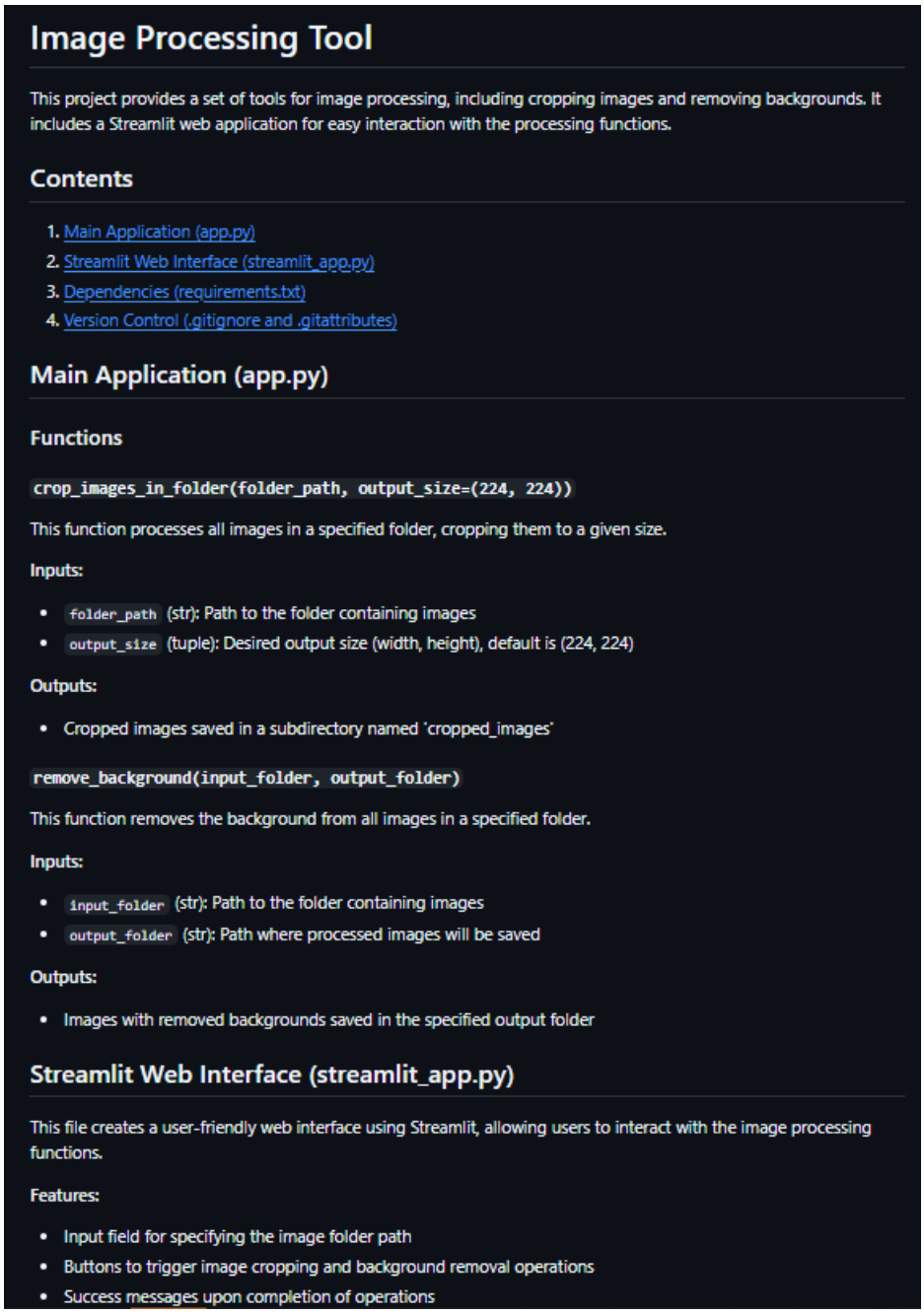


Image Processing Tool

This project provides a set of tools for image processing, including cropping images and removing backgrounds. It includes a Streamlit web application for easy interaction with the processing functions.

Contents

1. Main Application (app.py)
2. Streamlit Web Interface (streamlit_app.py)
3. Dependencies (requirements.txt)
4. Version Control (.gitignore and .gitattributes)

Main Application (app.py)

Functions

`crop_images_in_folder(folder_path, output_size=(224, 224))`

This function processes all images in a specified folder, cropping them to a given size.

Inputs:

- `folder_path` (str): Path to the folder containing images
- `output_size` (tuple): Desired output size (width, height), default is (224, 224)

Outputs:

- Cropped images saved in a subdirectory named 'cropped_images'

`remove_background(input_folder, output_folder)`

This function removes the background from all images in a specified folder.

Inputs:

- `input_folder` (str): Path to the folder containing images
- `output_folder` (str): Path where processed images will be saved

Outputs:

- Images with removed backgrounds saved in the specified output folder

Streamlit Web Interface (streamlit_app.py)

This file creates a user-friendly web interface using Streamlit, allowing users to interact with the image processing functions.

Features:

- Input field for specifying the image folder path
- Buttons to trigger image cropping and background removal operations
- Success messages upon completion of operations

Figura 1. Archivo README del repositorio de GitHub. (https://github.com/danielmedinam03/Preprocesado-imagenes-rembg)

En este proyecto se desarrolla una herramienta para procesamiento de imágenes que incluye funcionalidades como recortar imágenes y eliminar fondos. La herramienta está diseñada para facilitar el trabajo con lotes de imágenes mediante una aplicación web interactiva construida con Streamlit. Entre las principales características destacan funciones específicas para recortar imágenes a un tamaño determinado y eliminar fondos, con los resultados guardados automáticamente en carpetas organizadas. Este proyecto es especialmente útil para preparar conjuntos de datos para modelos de aprendizaje automático o para limpiar colecciones de imágenes, optimizando el flujo de trabajo para tareas relacionadas con visión por computadora.

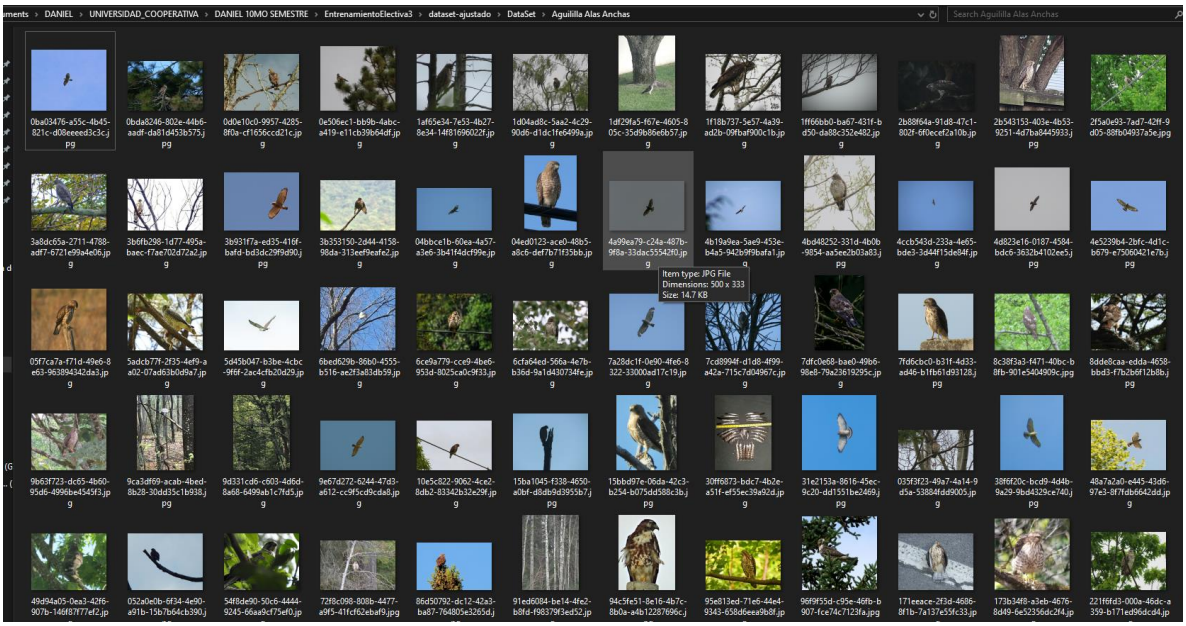


Figura 2. Imágenes descargadas.

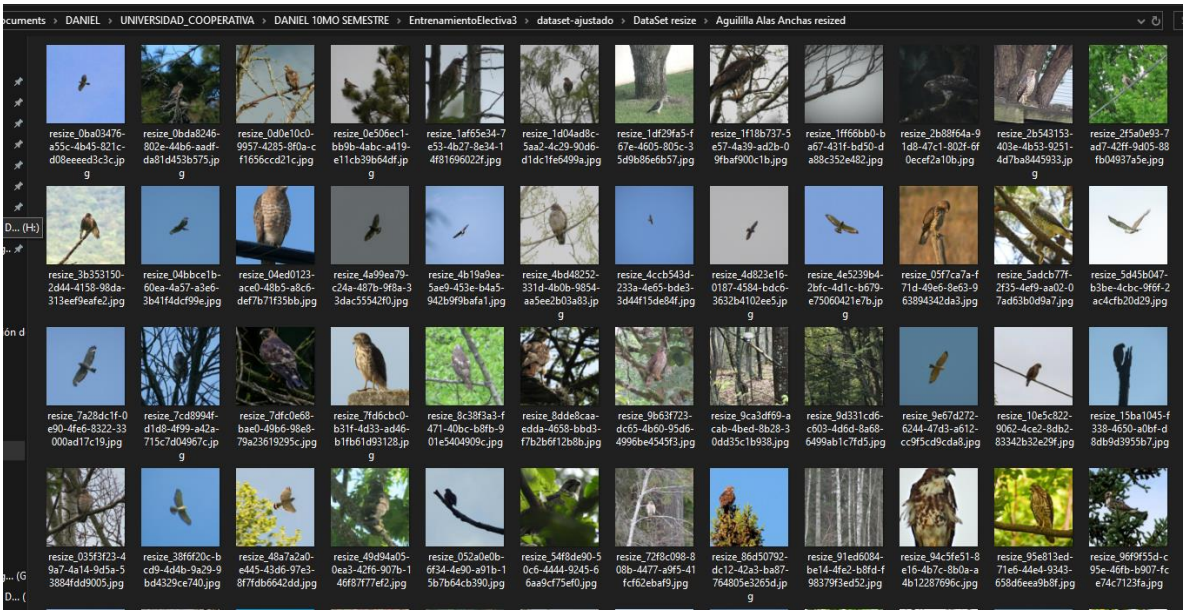


Figura 3. Imágenes redimensionadas

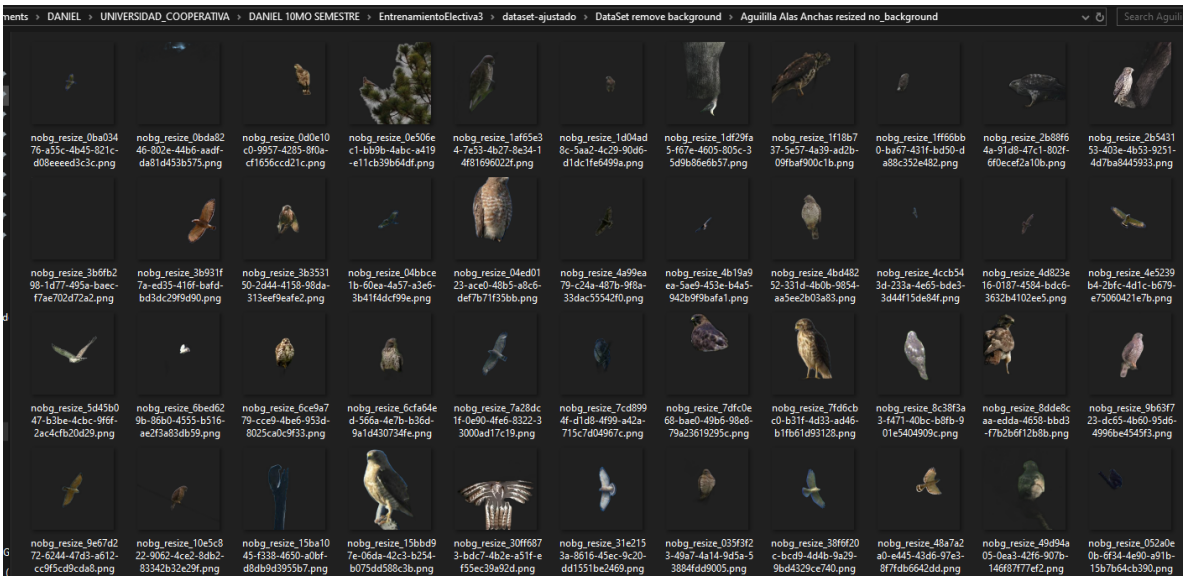


Figura 4. Imágenes sin fondo.

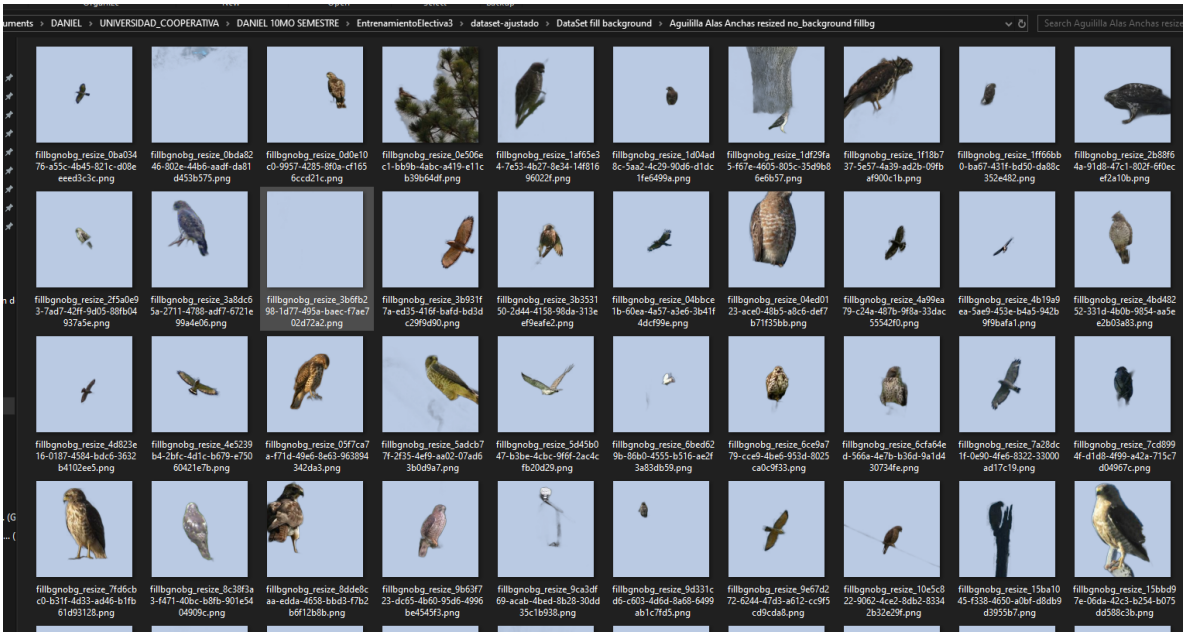


Figura 5. Imágenes con fondo azul

Modelado

Comprensión del negocio, compresión de los datos y preparación de los datos.

Aplicación de la Metodología CRISP-DM

Comprensión del Negocio:

El objetivo del proyecto es construir un modelo de clasificación de imágenes para identificar diferentes especies de aves a partir de imágenes.

Comprensión de los Datos:

Las imágenes de entrenamiento pertenecen a diversas especies de aves. Analizar la distribución de clases es importante para evitar sesgos. Como observamos en la matriz de confusión, algunas clases están mejor representadas que otras, lo cual afecta el rendimiento del modelo en identificar las especies menos comunes.

Las imagenes se obtuvieron del buscador de Google y de la app eBird

Preparación de los Datos:

Durante la fase de preparación se redimensionaron las imagenes, se recortaron, se les quito el fondo y luego se les aplico el fondo de color azul

Para el entrenamiento se usaron imagenes sin fondo

Se importan librerias requeridas

[1]:
```
from keras.models import Sequential, Model
from keras.layers import Conv2D, MaxPool2D, Dense, Flatten, Dropout, BatchNormalization, Input
from keras.optimizers import Adam
from keras.callbacks import TensorBoard, ModelCheckpoint
from keras.utils import to_categorical
import os
import numpy as np
from keras.preprocessing import image
from keras.applications.imagenet_utils import preprocess_input, decode_predictions
from keras.applications.vgg16 import VGG16
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import load_model
from sklearn.utils import shuffle
from sklearn.model_selection import train_test_split
import cv2
import matplotlib.pyplot as plt
%matplotlib inline
```

Construcción del modelo y modelado.

Construcción del modelo

Se definen los data set de entrenamiento y validacion

[2]:
```
train_data_dir = 'dataset-ajustado-150/train'
validation_data_dir = 'dataset-ajustado-150/valid'
```

[3]:
```
width_shape = 224
height_shape = 224
batch_size = 32
```

Se aplicaron técnicas de aumento de datos como rotación, zoom, desplazamiento, etc. Esto ayuda a mejorar la capacidad del modelo para generalizar. Sin embargo, podría ser necesario revisar si ciertas clases necesitan un aumento de datos específico (por ejemplo, si algunas especies tienen menos imágenes disponibles).Se definen las rutas de los datos y se establecen parámetros como el tamaño de imagen (224x224) y el tamaño de lote (32).

Modelado

Se utilizó el modelo VGG16, el cual es adecuado para la clasificación de imágenes complejas. Sin embargo, los resultados sugieren que tal vez sea necesario ajustar los hiperparámetros o incluso probar otros modelos como ResNet o EfficientNet para mejorar el rendimiento.

[4]:
```
# Definir el generador de imágenes para el conjunto de entrenamiento con aumentos de datos
train_datagen = ImageDataGenerator(
    rotation_range=20,              # Rango de grados para rotación aleatoria
    zoom_range=0.2,                 # Rango de zoom aleatorio
    width_shift_range=0.1,          # Rango de desplazamiento horizontal aleatorio
    height_shift_range=0.1,         # Rango de desplazamiento vertical aleatorio
    horizontal_flip=True,           # Volteo horizontal aleatorio
    vertical_flip=False,            # No se aplica volteo vertical
    preprocessing_function=preprocess_input)  # Función de preprocesamiento

# Definir el generador de imágenes para el conjunto de validación con los mismos aumentos de datos
valid_datagen = ImageDataGenerator(
    rotation_range=20,
    zoom_range=0.2,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True,
    vertical_flip=False,
    preprocessing_function=preprocess_input)

# Crear un generador de lotes de imágenes para el conjunto de entrenamiento
train_generator = train_datagen.flow_from_directory(
    train_data_dir,                 # Directorio que contiene las imágenes de entrenamiento
```

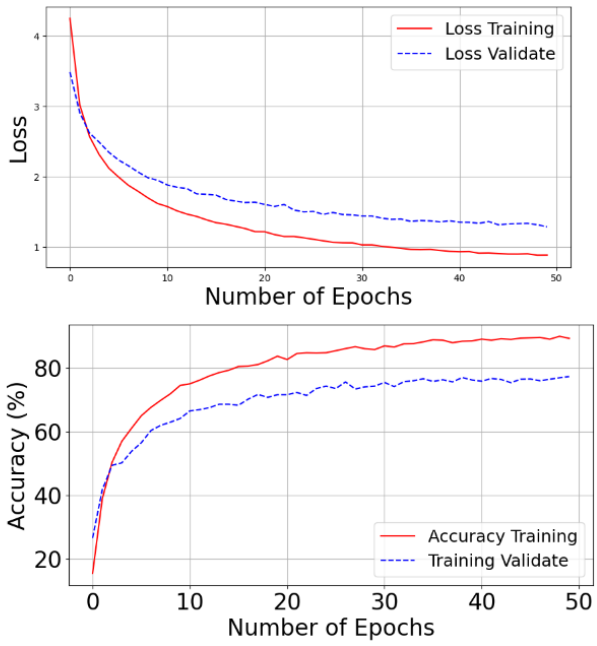
Configuracion del modelo.

Se configura un modelo de clasificación de imágenes usando VGG16. Son 5949 muestras para entrenamiento, 1299 para validacion, 44 clases y 50 épocas para el entrenamiento.

[5]:
```
# Definir el número de muestras de entrenamiento y validación
nb_train_samples = 21786
nb_validation_samples = 6225
num_classes = 44  # Cambia este valor según el número de clases en tu conjunto de datos
epochs = 50  # Cambia este valor según tus necesidades

# Definir la entrada de la red neuronal con el tamaño de las imágenes
image_input = Input(shape=(width_shape, height_shape, 3))

# Cargar el modelo VGG16 preentrenado con pesos ajustados desde ImageNet
model = VGG16(input_tensor=image_input, include_top=True, weights='imagenet')

# Obtener la salida de la penúltima capa densa del modelo VGG16 (fc2)
last_layer = model.get_layer('fc2').output

# Añadir una nueva capa densa al final del modelo para la clasificación multiclase con regularización L2 (Evita sobreajuste)
out = Dense(num_classes, activation='softmax', kernel_regularizer='l2', name='output')(last_layer)

# Crear un nuevo modelo personalizado que toma la entrada de la imagen y produce la salida clasificada
custom_vgg_model = Model(image_input, out)

# Congelar todas las capas del modelo, excepto la capa densa añadida
for layer in custom_vgg_model.layers[:-1]:
    layer.trainable = False

# Compilar el modelo con una función de pérdida, optimizador y métricas especificadas
custom_vgg_model.compile(loss='categorical_crossentropy', optimizer=Adam(learning_rate=0.0001), metrics=['accuracy'])


# Mostrar un resumen del modelo que incluye la arquitectura y el número de parámetros
custom_vgg_model.summary()

# Entrenar el modelo utilizando generadores de datos para el conjunto de entrenamiento y validación
model_history = custom_vgg_model.fit(
    train_generator,
    epochs=epochs,
    validation_data=validation_generator,
    steps_per_epoch=nb_train_samples//batch_size,  # Número de pasos por época de entrenamiento
    validation_steps=nb_validation_samples//batch_size)  # Número de pasos por época de validación
```

En el primer gráfico de pérdida, el hecho de que las pérdidas disminuyan con las épocas es positivo, pero la diferencia entre las pérdidas de entrenamiento y validación puede sugerir sobreajuste.

En el segundo gráfico de precisión, la brecha entre las precisiones de entrenamiento y validación también indica que el modelo se está adaptando demasiado a los datos de entrenamiento y no generaliza tan bien a los datos de validación.



Evaluación

- El modelo no acertó, ¿Por qué?
- Datos insuficientes o desbalanceados
- Características similares entre clases
- Limitaciones del modelo
- Errores en el preprocesamiento
- Ambigüedad en los datos

```
# Leer la imagen, cambiar tamaño y preprocesar
imaget=cv2.resize(cv2.imread(imaget_path), (width_shape, height_shape), interpolation = cv2.INTER_AREA)
xt = np.asarray(imaget)
xt=preprocess_input(xt)
xt = np.expand_dims(xt,axis=0)

# Obtener las predicciones del modelo
preds = modelt.predict(xt)

# Obtener la clase predicha y su porcentaje de confianza
predicted_class_index = np.argmax(preds)
predicted_class_name = names[predicted_class_index]
confidence_percentage = preds[0][predicted_class_index] * 100

# Imprimir el resultado
print(f'Clase predicha: {predicted_class_name}')
print(f'Porcentaje de confianza: {confidence_percentage:.2f}%')

# Mostrar la imagen
plt.imshow(cv2.cvtColor(np.asarray(imaget), cv2.COLOR_BGR2RGB))
plt.axis('off')
plt.show()
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 553ms/step
Clase predicha: buho cara oscura
Porcentaje de confianza: 32.07%
```

Métricas del modelo con cada una de las clases

```
print(metrics.classification_report(y_real,y_pred, digits = 4))
```

```
Found 650 images belonging to 44 classes.
21/21 ━━━━━━━━━━━━━━━━━━━━ 95s 5s/step
              precision    recall  f1-score   support

           0     0.1176    0.2105    0.1509        19
           1     0.0833    0.0909    0.0870        11
           2     0.5000    0.0667    0.1176        15
           3     0.5000    0.1579    0.2400        19
           4     0.6364    0.5833    0.6087        12
           5     0.6000    0.3333    0.4286        18
           6     0.7500    0.3333    0.4615         9
           7     0.0909    0.3333    0.1429        15
           8     0.6429    0.5625    0.6000        16
           9     0.2609    0.7500    0.3871        16
          10     0.6667    0.1333    0.2222        15
          11     0.2500    0.2667    0.2581        15
          12     0.2000    0.0435    0.0714        23
          13     1.0000    0.1176    0.2105        17
          14     0.6667    0.2000    0.3077        10
          15     0.5000    0.3125    0.3846        16
          16     0.5714    0.3077    0.4000        13
          17     0.2667    0.8571    0.4068        14
          18     0.1765    0.1765    0.1765        17
          19     0.3529    0.4286    0.3871        14
          20     0.5000    0.2500    0.3333        12
          21     0.7500    0.2308    0.3529        13
          22     0.7692    0.3846    0.5128        26
          23     0.2424    0.5714    0.3404        14
          24     0.3200    0.8889    0.4706         9
          25     0.3529    0.6667    0.4615        18
          26     0.3750    0.4737    0.4186        19
          27     0.0000    0.0000    0.0000         9
          28     0.8889    0.6154    0.7273        13
          29     1.0000    0.2143    0.3529        14
          30     0.4000    0.4615    0.4286        13
          31     1.0000    0.2667    0.4211        15
          32     1.0000    0.2727    0.4286        11
          33     0.4062    0.8667    0.5532        15
          34     0.3333    0.8182    0.4737        11
          35     1.0000    0.3684    0.5385        19
          36     0.7500    0.2727    0.4000        11
          37     0.9091    0.8333    0.8696        12
          38     1.0000    0.3636    0.5333        11
          39     0.5455    0.3000    0.3871        20
          40     0.3333    0.0667    0.1111        15
          41     0.1429    0.1176    0.1290        17
          42     0.4643    0.8125    0.5909        16
          43     0.5294    0.6923    0.6000        13

    accuracy                         0.3785       650
   macro avg     0.5192    0.3880    0.3746       650
weighted avg     0.5140    0.3785    0.3688       650
```

El modelo presenta una precisión general del 51.92% y un F1-score promedio ponderado de 0.3688, reflejando un desempeño moderado. Sin embargo, existe una alta variabilidad entre clases, con algunas alcanzando buenos resultados mientras que otras muestran valores muy bajos de precisión y recall, lo que sugiere un desbalance en los datos o dificultades con clases específicas. Se recomienda aplicar técnicas como aumento de datos, ajustes en la función de pérdida y análisis detallado de la matriz de confusión para mejorar la representación de clases menos precisas y optimizar el rendimiento general del modelo.

Conclusiones

El proyecto demuestra el potencial del uso de técnicas avanzadas de aprendizaje profundo, como Transfer Learning con arquitecturas preentrenadas como ResNet50 y VGG16, en combinación con un preprocesamiento efectivo de imágenes, para desarrollar un modelo de clasificación de aves. A pesar de lograr una precisión general del 51.92%, los resultados muestran una alta variabilidad entre clases, lo que sugiere la necesidad de abordar problemas de desbalance en los datos y optimizar la representación de las clases menos precisas. El preprocesamiento, que incluyó la eliminación de fondos y el recorte de imágenes, fue clave para mejorar la calidad del conjunto de datos, aunque posibles errores en este proceso podrían haber influido en el bajo desempeño de algunas clases. La herramienta desarrollada con Streamlit facilitó el manejo y preparación de imágenes, lo que representa una fortaleza para proyectos futuros. Sin embargo, el modelo no es completamente confiable para aplicaciones en tiempo real debido a su desempeño inconsistente, lo que resalta la importancia de implementar estrategias adicionales, como el aumento de datos, el ajuste de la función de pérdida y un análisis más profundo de las clases con bajo rendimiento. En conclusión, este proyecto sienta las bases para una solución funcional, pero requiere mejoras para alcanzar un rendimiento más robusto y uniforme en la clasificación de aves.