

07MIAR - Redes Neuronales y Deep Learning: Proyecto de programación *"Deep Vision in classification tasks"*

Grupo LUYOANKA

Miembros del equipo:

Luisa Fernanda Alarcón Vargas

Yovanny Enrique Alvarado Pérez

Karolin Yessenia Castañeda Vásquez

Ana María Vera Martín

Disclaimer: Este proyecto puede contener información parcial o fracciones de código generado con la IA.

Enunciado

En esta actividad, el alumno debe **evaluar y comparar dos estrategias** para la **clasificación de imágenes** empleando el **dataset asignado**. Los alumnos deberá resolver el reto proponiendo una solución válida **basada en aprendizaje profundo**, más concretamente en redes neuronales convolucionales (**CNNs**). Será indispensable que la solución propuesta siga el **pipeline visto en clase** para resolver este tipo de tareas de inteligencia artificial:

1. **Carga** del conjunto de datos
2. **Inspección** del conjunto de datos
3. **Acondicionamiento** del conjunto de datos
4. Desarrollo de la **arquitectura** de red neuronal y **entrenamiento** de la solución
5. **Monitorización** del proceso de **entrenamiento** para la toma de decisiones
6. **Evaluación** del modelo predictivo y planteamiento de la siguiente prueba experimental

****Estrategia 1: Entrenar desde cero o *from scratch*****

La primera estrategia a comparar será una **red neuronal profunda** que el **alumno debe diseñar, entrenar y optimizar**. Se debe **justificar empíricamente** las decisiones que llevaron a la selección de la **arquitectura e hiperparámetros final**. Se espera que el alumno utilice todas las **técnicas de regularización** mostradas en clase de forma justificada para la mejora del rendimiento de la red neuronal

(*weight regularization, dropout, batch normalization, data augmentation, etc.*).

Estrategia 2: Red pre-entrenada

La segunda estrategia a comparar debe incluir la utilización de una **red preentrenada** con el dataset ImageNet, llevando a cabo tareas de **transfer learning** y **fine-tuning** para resolver la tarea de clasificación asignada. Deben **compararse al menos dos tipos de arquitecturas** (VGGs, ResNet50, Xception, InceptionV3, InceptionResNetV2, MobileNetV2, DenseNet, ResNet) y se debe **seleccionar la que mayor precisión proporcione** (información sobre las arquitecturas disponibles en <https://keras.io/applications/>). Se espera el uso de todas las **técnicas de regularización** mostradas en clase de forma justificada para la mejora del rendimiento de la red neuronal (*weight regularization, dropout, batch normalization, data augmentation, etc.*).

Preparacion del entorno

Vamos a emplear una carpeta alojada en Google Drive que se llama DL y en ella vamos a trabajar todo

```
In [ ]: !pip install -q gdown
```

```
# Descargar la carpeta entera usando gdown (necesita el ID)
!gdown --folder https://drive.google.com/drive/folders/11Y7UGXLqSYk3stbQuF17
```

```
Retrieving folder contents
Processing file 1HZ51uTdK8Ds5-ITtIwZGHp0M4f1EVp6c 07MIAR_Proyecto_Programacion.ipynb
Processing file 1xFc8AMT58mn0p102h24BpFN0u_xm0mVc kaggle.json
Retrieving folder contents completed
Building directory structure
Building directory structure completed
Downloading...
From: https://drive.google.com/uc?id=1HZ51uTdK8Ds5-ITtIwZGHp0M4f1EVp6c
To: /content/DL/07MIAR_Proyecto_Programacion.ipynb
100% 2.68M/2.68M [00:00<00:00, 160MB/s]
Downloading...
From: https://drive.google.com/uc?id=1xFc8AMT58mn0p102h24BpFN0u_xm0mVc
To: /content/DL/kaggle.json
100% 69.0/69.0 [00:00<00:00, 298kB/s]
Download completed
```

```
In [ ]: # Establezco una ruta absoluta a un directorio existente de mi Google Drive
BASE_FOLDER = "DL"
```

Carga del Dataset desde Kaggle

En el directorio de trabajo tenemos almacenado el fichero kaggle.json para poder descargarnos el dataset de kaggle.

```
In [ ]: # Nos aseguramos que tenemos instalada la última versión API de Kaggle
!pip install --upgrade --force-reinstall --no-deps kaggle
```

```
Collecting kaggle
  Downloading kaggle-1.7.4.5-py3-none-any.whl.metadata (16 kB)
  Downloading kaggle-1.7.4.5-py3-none-any.whl (181 kB)
    0.0/181.2 kB ? eta -:-:-
    181.2/181.2 kB 4.2 MB/s eta 0:00:00
Installing collected packages: kaggle
  Attempting uninstall: kaggle
    Found existing installation: kaggle 1.7.4.5
    Uninstalling kaggle-1.7.4.5:
      Successfully uninstalled kaggle-1.7.4.5
  Successfully installed kaggle-1.7.4.5
```

```
In [ ]: #Importacion de librerias
import os
import shutil
import zipfile
from collections import defaultdict, Counter
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from PIL import Image
import json
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import numpy as np
```

```
In [ ]: #Para asegurarnos que la ruta existe
os.makedirs(BASE_FOLDER, exist_ok=True)

# Path del fichero kaggle en la carpeta BASE_FOLDER
kaggle_json_path = os.path.join(BASE_FOLDER, "kaggle.json")

# Mover kaggle.json a la ubicación requerida
os.makedirs(os.path.expanduser("~/kaggle"), exist_ok=True)
shutil.copy(kaggle_json_path, os.path.expanduser("~/kaggle/kaggle.json"))
os.chmod(os.path.expanduser("~/kaggle/kaggle.json"), 0o600)

# Descargar el dataset al BASE_FOLDER
!kaggle datasets download -d tr1gg3rtrash/yoga-posture-dataset -p "$BASE_FOL

# Descomprimir el archivo ZIP descargado en una subcarpeta
zip_path = os.path.join(BASE_FOLDER, "yoga-posture-dataset.zip")
extract_path = os.path.join(BASE_FOLDER, "yoga_dataset")
```

```
os.makedirs(extract_path, exist_ok=True)
with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    zip_ref.extractall(extract_path)

print(f"Dataset extraído en: {extract_path}")
```

Dataset URL: <https://www.kaggle.com/datasets/tr1gg3rtrash/yoga-posture-dataset>

License(s): CC0-1.0

Downloading yoga-posture-dataset.zip to DL

99% 443M/447M [00:02<00:00, 128MB/s]

100% 447M/447M [00:02<00:00, 186MB/s]

Dataset extraído en: DL/yoga_dataset

```
In [ ]: #Comprobamos que tenemos datos en la carpeta y se ha descomprimido ok
!find "$BASE_FOLDER/yoga_dataset" -type f | head -n 10
```

```
DL/yoga_dataset/Utthita Hasta Padangusthasana/File37.png
DL/yoga_dataset/Utthita Hasta Padangusthasana/File8.png
DL/yoga_dataset/Utthita Hasta Padangusthasana/File38.png
DL/yoga_dataset/Utthita Hasta Padangusthasana/File17.png
DL/yoga_dataset/Utthita Hasta Padangusthasana/File11.png
DL/yoga_dataset/Utthita Hasta Padangusthasana/File2.png
DL/yoga_dataset/Utthita Hasta Padangusthasana/File60.jpeg
DL/yoga_dataset/Utthita Hasta Padangusthasana/File49.png
DL/yoga_dataset/Utthita Hasta Padangusthasana/File19.png
DL/yoga_dataset/Utthita Hasta Padangusthasana/File51.png
```

```
In [ ]: # Creamos la ruta donde se encuentran los datos descomprimidos y
#comprobamos que accedemos a ellos
folder_path = os.path.join(BASE_FOLDER, "yoga_dataset")
print(f"Ruta completa a la carpeta: {folder_path}")
!find "{folder_path}" -type f | head -n 10
```

```
Ruta completa a la carpeta: DL/yoga_dataset
DL/yoga_dataset/Utthita Hasta Padangusthasana/File37.png
DL/yoga_dataset/Utthita Hasta Padangusthasana/File8.png
DL/yoga_dataset/Utthita Hasta Padangusthasana/File38.png
DL/yoga_dataset/Utthita Hasta Padangusthasana/File17.png
DL/yoga_dataset/Utthita Hasta Padangusthasana/File11.png
DL/yoga_dataset/Utthita Hasta Padangusthasana/File2.png
DL/yoga_dataset/Utthita Hasta Padangusthasana/File60.jpeg
DL/yoga_dataset/Utthita Hasta Padangusthasana/File49.png
DL/yoga_dataset/Utthita Hasta Padangusthasana/File19.png
DL/yoga_dataset/Utthita Hasta Padangusthasana/File51.png
```

Inspeccion del dataset

Vamos a analizar el dataset. Para ello se hará lo siguiente:

1. Contar el numero de imagenes y comprobar su estructura.
2. Visualizar distribuciones estadísticas.
3. Mostrar una imagen por cada clase del dataset.

4. Verificar el tamaño de las imágenes
5. Verificar si hay imágenes corruptas
6. Analizar el formato y tipo de imagen
7. Analizar si existen etiquetas o metadata

Conteo del numero de imagenes y comprobacion de estructura

```
In [ ]: def revisar_estructura_y_contar_imagenes(base_path):  
    class_counts = {}  
  
    for clase in os.listdir(base_path):  
        ruta_clase = os.path.join(base_path, clase)  
        if os.path.isdir(ruta_clase):  
            imagenes = [f for f in os.listdir(ruta_clase) if f.lower().endsw  
            num_imagenes = len(imagenes)  
            class_counts[clase] = num_imagenes  
            print(f"Clase: {clase}, Imágenes: {num_imagenes}")  
        else:  
            print(f"{ruta_clase} no es una carpeta")  
  
    # Mostrar todas las clases y su respectivo número de imágenes  
    for class_name, count in sorted(class_counts.items()):  
  
        return class_counts  
  
# Llamada a la función para inspeccionar el dataset  
class_counts = revisar_estructura_y_contar_imagenes(folder_path)
```

Clase: Marjaryasana, Imágenes: 56
Clase: Uttanasana, Imágenes: 71
Clase: Baddha Konasana, Imágenes: 81
Clase: Ardha Matsyendrasana, Imágenes: 96
Clase: Adho Mukha Svanasana, Imágenes: 74
Clase: Phalakasana, Imágenes: 66
Clase: Virabhadrasana Three, Imágenes: 69
Clase: Camatkarasana, Imágenes: 62
Clase: Parsva Virabhadrasana, Imágenes: 14
Clase: Hanumanasana, Imágenes: 41
Clase: Setu Bandha Sarvangasana, Imágenes: 66
Clase: Malasana, Imágenes: 73
Clase: Sivasana, Imágenes: 20
Clase: Utthita Hasta Padangusthasana, Imágenes: 64
Clase: Bakasana, Imágenes: 84
Clase: Vasisthasana, Imágenes: 80
Clase: Anjaneyasana, Imágenes: 71
Clase: Supta Kapotasana, Imágenes: 13
Clase: Salamba Sarvangasana, Imágenes: 73
Clase: Ardha Navasana, Imágenes: 13
Clase: Virabhadrasana Two, Imágenes: 60
Clase: Upavistha Konasana, Imágenes: 17
Clase: Pincha Mayurasana, Imágenes: 43
Clase: Padmasana, Imágenes: 77
Clase: Utthita Parsvakonasana, Imágenes: 69
Clase: Urdhva Mukha Svsnssana, Imágenes: 69
DL/yoga_dataset/Poses.json no es una carpeta
Clase: Balasana, Imágenes: 79
Clase: Bitilasana, Imágenes: 94
Clase: Trikonasana, Imágenes: 23
Clase: Eka Pada Rajakapotasana, Imágenes: 53
Clase: Navasana, Imágenes: 18
Clase: Ardha Chandrasana, Imágenes: 59
Clase: Ashta Chandrasana, Imágenes: 12
Clase: Salamba Bhujangasana, Imágenes: 62
Clase: Adho Mukha Vrksasana, Imágenes: 65
Clase: Ardha Pincha Mayurasana, Imágenes: 54
Clase: Dhanurasana, Imágenes: 54
Clase: Virabhadrasana One, Imágenes: 63
Clase: Utkatasana, Imágenes: 81
Clase: Urdhva Dhanurasana, Imágenes: 74
Clase: Halasana, Imágenes: 71
Clase: Ustrasana, Imágenes: 96
Clase: Alanasana, Imágenes: 18
Clase: Parsvottanasana, Imágenes: 43
Clase: Vrksasana, Imágenes: 68
Clase: Garudasana, Imágenes: 85
Clase: Paschimottanasana, Imágenes: 62

Observamos que hay 53 clases y un archivo que no es una clase. Las clases se ven desbalanceadas, porque nos encontramos con algunas clases que tienen decenas de imágenes frente a otras que tiene mas de 80.

Visualizar distribuciones estadísticas

```
In [ ]: # Convertimos a DataFrame
df = pd.DataFrame(list(class_counts.items()), columns=["Clase", "Número de I

# Ordenamos por número de imágenes
df = df.sort_values("Número de Imágenes", ascending=True)

# Gráfico de barras horizontales para una mejor visualización
plt.figure(figsize=(10, len(df) * 0.2))
ax = sns.barplot(data=df, x="Número de Imágenes", y="Clase", palette="viridi

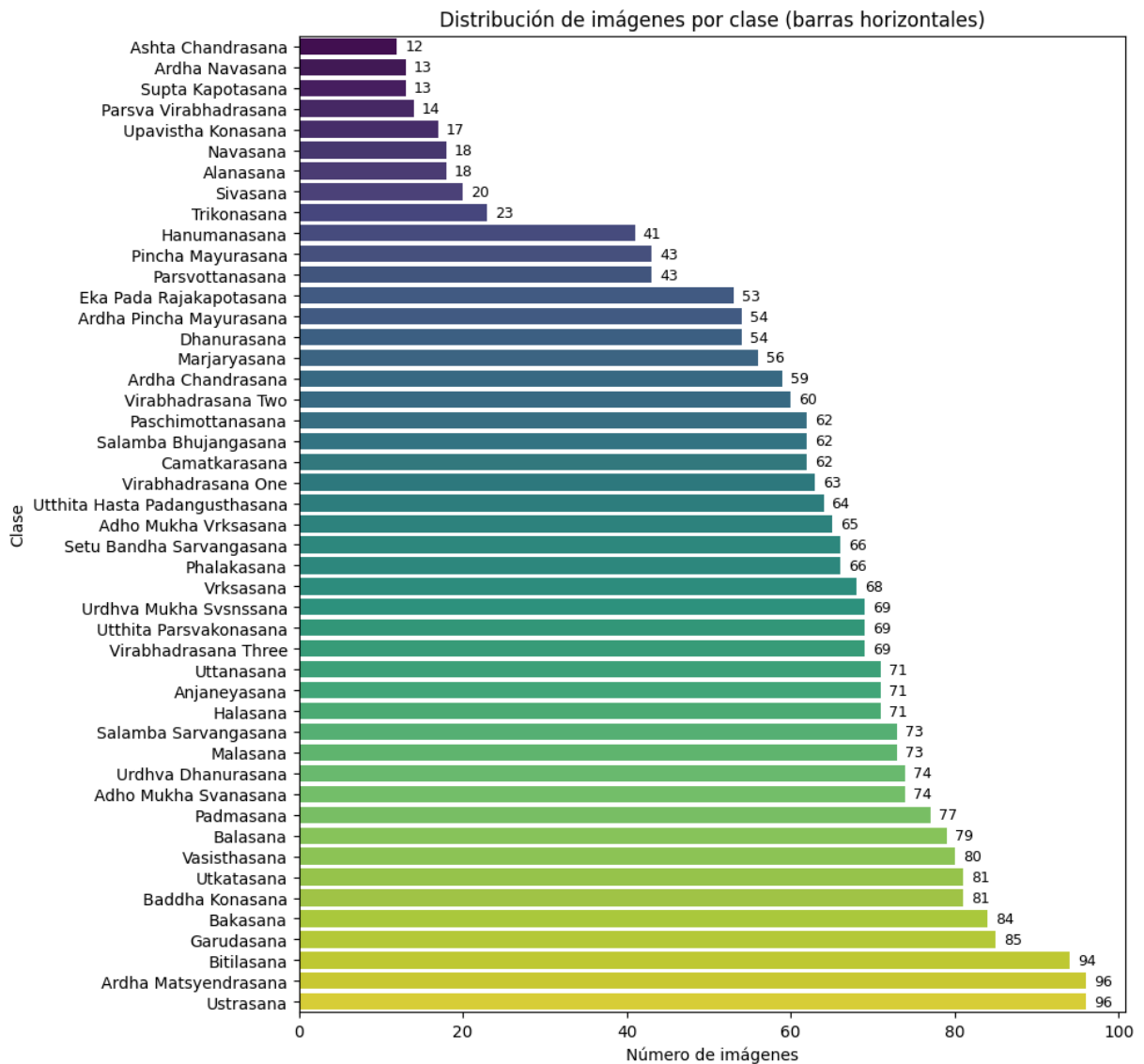
# Mostramos el número de muestras que tenemos de cada clase
for i, value in enumerate(df["Número de Imágenes"]):
    ax.text(value + 1, i, str(value), va='center', fontsize=9)

plt.title("Distribución de imágenes por clase (barras horizontales)")
plt.xlabel("Número de imágenes")
plt.ylabel("Clase")
plt.tight_layout()
plt.show()
```

<ipython-input-9-51508f53c01b>:9: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

```
ax = sns.barplot(data=df, x="Número de Imágenes", y="Clase", palette="viridis")
```



En la imagen podemos observar el desbalanceo de clases que estábamos anticipando en el punto anterior.

Mostramos una imagen por cada clase del dataset.

```
In [ ]: def mostrar_imagenes_por_clase(base_path, clases, n=3): # Cambiar n a 3 para
total_clases = len(clases)
filas = (total_clases + n - 1) // n # Calcular el número de filas neces

plt.figure(figsize=(n * 3, filas * 3)) # Ajustar el tamaño de la figura

for i, clase in enumerate(clases):
    ruta_clase = os.path.join(base_path, clase)
    # Verificar que es un directorio (carpeta) y no un archivo
    if os.path.isdir(ruta_clase):
        imagenes = [f for f in os.listdir(ruta_clase) if f.lower().endsw
        if imagenes:
            img_file = imagenes[0] # Tomar solo la primera imagen
```



```

img_path = os.path.join(ruta_clase, img_file)
img = mpimg.imread(img_path)

# Organizar las imágenes en filas y columnas (3 imágenes por
plt.subplot(filas, n, i + 1)
plt.imshow(img)
plt.title(clase)
plt.axis('off')

plt.tight_layout()

plt.savefig("/content/imagenes_dataset.png") # Guarda la imagen en Drive

plt.show()

# Obtener todas las clases (carpetas) presentes en la ruta base
clases = [c for c in os.listdir(folder_path) if os.path.isdir(os.path.join(f

# Llamar a la función para mostrar una imagen por clase
mostrar_imagenes_por_clase(folder_path, clases, n=3)

```

In []: `from IPython.display import display, Image`

```

# Mostrar imagen primero
display(Image(filename="/content/imagenes_dataset.png"))

```

Output hidden; open in <https://colab.research.google.com> to view.

Verificar el tamaño de las imágenes

```

In [ ]: def verificar_tamano_imagenes(base_path):
    for clase in os.listdir(base_path):
        ruta_clase = os.path.join(base_path, clase)
        if os.path.isdir(ruta_clase):
            imagenes = [f for f in os.listdir(ruta_clase) if f.lower().endsw
            for img_file in imagenes[:5]: # Comprobar solo las primeras 5 i
            img_path = os.path.join(ruta_clase, img_file)
            with Image.open(img_path) as img:
                print(f"Clase: {clase}, Imagen: {img_file}, Tamaño: {img

verificar_tamano_imagenes(folder_path)

```

Clase: Marjaryasana, Imagen: File16.png, Tamaño: (515, 309)
Clase: Marjaryasana, Imagen: File17.png, Tamaño: (433, 258)
Clase: Marjaryasana, Imagen: File29.png, Tamaño: (473, 294)
Clase: Marjaryasana, Imagen: File51.jpeg, Tamaño: (275, 183)
Clase: Marjaryasana, Imagen: File28.png, Tamaño: (531, 307)
Clase: Uttanasana, Imagen: File16.png, Tamaño: (235, 481)
Clase: Uttanasana, Imagen: File17.png, Tamaño: (340, 642)
Clase: Uttanasana, Imagen: File29.png, Tamaño: (125, 273)
Clase: Uttanasana, Imagen: File45.png, Tamaño: (217, 411)
Clase: Uttanasana, Imagen: File56.png, Tamaño: (184, 329)
Clase: Baddha Konasana, Imagen: File16.png, Tamaño: (405, 414)
Clase: Baddha Konasana, Imagen: File49.jpg, Tamaño: (577, 509)
Clase: Baddha Konasana, Imagen: File17.png, Tamaño: (416, 393)
Clase: Baddha Konasana, Imagen: File29.png, Tamaño: (481, 310)
Clase: Baddha Konasana, Imagen: File28.png, Tamaño: (374, 394)
Clase: Ardha Matsyendrasana, Imagen: File16.png, Tamaño: (270, 282)
Clase: Ardha Matsyendrasana, Imagen: File17.png, Tamaño: (379, 275)
Clase: Ardha Matsyendrasana, Imagen: File29.png, Tamaño: (500, 366)
Clase: Ardha Matsyendrasana, Imagen: File71.png, Tamaño: (623, 770)
Clase: Ardha Matsyendrasana, Imagen: File28.png, Tamaño: (377, 466)
Clase: Adho Mukha Svanasana, Imagen: File16.png, Tamaño: (511, 309)
Clase: Adho Mukha Svanasana, Imagen: File17.png, Tamaño: (731, 436)
Clase: Adho Mukha Svanasana, Imagen: File29.png, Tamaño: (359, 392)
Clase: Adho Mukha Svanasana, Imagen: File28.png, Tamaño: (658, 450)
Clase: Adho Mukha Svanasana, Imagen: File73.jpeg, Tamaño: (275, 183)
Clase: Phalakasana, Imagen: File16.png, Tamaño: (765, 703)
Clase: Phalakasana, Imagen: File17.png, Tamaño: (388, 211)
Clase: Phalakasana, Imagen: File29.png, Tamaño: (1120, 496)
Clase: Phalakasana, Imagen: File28.png, Tamaño: (242, 158)
Clase: Phalakasana, Imagen: File45.png, Tamaño: (536, 324)
Clase: Virabhadrasana Three, Imagen: File16.png, Tamaño: (55, 122)
Clase: Virabhadrasana Three, Imagen: File17.png, Tamaño: (106, 124)
Clase: Virabhadrasana Three, Imagen: File29.png, Tamaño: (715, 586)
Clase: Virabhadrasana Three, Imagen: File28.png, Tamaño: (527, 278)
Clase: Virabhadrasana Three, Imagen: File45.png, Tamaño: (852, 480)
Clase: Camatkarasana, Imagen: File16.png, Tamaño: (792, 469)
Clase: Camatkarasana, Imagen: File17.png, Tamaño: (533, 317)
Clase: Camatkarasana, Imagen: File29.png, Tamaño: (479, 252)
Clase: Camatkarasana, Imagen: File28.png, Tamaño: (597, 453)
Clase: Camatkarasana, Imagen: File45.png, Tamaño: (599, 231)
Clase: Parsva Virabhadrasana, Imagen: File13.jpeg, Tamaño: (183, 275)
Clase: Parsva Virabhadrasana, Imagen: File4.jpeg, Tamaño: (266, 189)
Clase: Parsva Virabhadrasana, Imagen: File2.jpeg, Tamaño: (246, 204)
Clase: Parsva Virabhadrasana, Imagen: File3.jpeg, Tamaño: (264, 191)
Clase: Parsva Virabhadrasana, Imagen: File11.jpeg, Tamaño: (275, 183)
Clase: Hanumanasana, Imagen: File16.png, Tamaño: (518, 336)
Clase: Hanumanasana, Imagen: File17.png, Tamaño: (368, 197)
Clase: Hanumanasana, Imagen: File29.png, Tamaño: (493, 402)
Clase: Hanumanasana, Imagen: File28.png, Tamaño: (461, 322)
Clase: Hanumanasana, Imagen: File15.png, Tamaño: (995, 627)
Clase: Setu Bandha Sarvangasana, Imagen: File16.png, Tamaño: (315, 131)
Clase: Setu Bandha Sarvangasana, Imagen: File49.jpg, Tamaño: (554, 231)
Clase: Setu Bandha Sarvangasana, Imagen: File17.png, Tamaño: (3949, 2306)
Clase: Setu Bandha Sarvangasana, Imagen: File29.png, Tamaño: (849, 685)
Clase: Setu Bandha Sarvangasana, Imagen: File28.png, Tamaño: (819, 379)
Clase: Malasana, Imagen: File16.png, Tamaño: (138, 255)

Clase: Malasana, Imagen: File17.png, Tamaño: (567, 806)
Clase: Malasana, Imagen: File29.png, Tamaño: (254, 323)
Clase: Malasana, Imagen: File28.png, Tamaño: (437, 565)
Clase: Malasana, Imagen: File73.jpeg, Tamaño: (299, 169)
Clase: Sivasana, Imagen: File13.jpeg, Tamaño: (329, 153)
Clase: Sivasana, Imagen: File4.jpeg, Tamaño: (275, 183)
Clase: Sivasana, Imagen: File3.jpeg, Tamaño: (286, 176)
Clase: Sivasana, Imagen: File11.jpeg, Tamaño: (275, 183)
Clase: Sivasana, Imagen: File20.jpeg, Tamaño: (275, 183)
Clase: Utthita Hasta Padangusthasana, Imagen: File16.png, Tamaño: (248, 450)
Clase: Utthita Hasta Padangusthasana, Imagen: File17.png, Tamaño: (249, 443)
Clase: Utthita Hasta Padangusthasana, Imagen: File29.png, Tamaño: (397, 389)
Clase: Utthita Hasta Padangusthasana, Imagen: File28.png, Tamaño: (177, 556)
Clase: Utthita Hasta Padangusthasana, Imagen: File45.png, Tamaño: (188, 250)
Clase: Bakasana, Imagen: File16.png, Tamaño: (573, 483)
Clase: Bakasana, Imagen: File17.png, Tamaño: (834, 370)
Clase: Bakasana, Imagen: File29.png, Tamaño: (318, 293)
Clase: Bakasana, Imagen: File71.png, Tamaño: (387, 365)
Clase: Bakasana, Imagen: File28.png, Tamaño: (308, 297)
Clase: Vasisthasana, Imagen: File16.png, Tamaño: (377, 304)
Clase: Vasisthasana, Imagen: File17.png, Tamaño: (768, 809)
Clase: Vasisthasana, Imagen: File29.png, Tamaño: (576, 569)
Clase: Vasisthasana, Imagen: File71.png, Tamaño: (278, 418)
Clase: Vasisthasana, Imagen: File28.png, Tamaño: (350, 344)
Clase: Anjaneyasana, Imagen: File16.png, Tamaño: (451, 428)
Clase: Anjaneyasana, Imagen: File17.png, Tamaño: (352, 248)
Clase: Anjaneyasana, Imagen: File29.png, Tamaño: (616, 723)
Clase: Anjaneyasana, Imagen: File28.png, Tamaño: (292, 372)
Clase: Anjaneyasana, Imagen: File45.png, Tamaño: (178, 281)
Clase: Supta Kapotasana, Imagen: File13.jpeg, Tamaño: (266, 190)
Clase: Supta Kapotasana, Imagen: File4.jpeg, Tamaño: (275, 183)
Clase: Supta Kapotasana, Imagen: File3.jpeg, Tamaño: (301, 168)
Clase: Supta Kapotasana, Imagen: File11.jpeg, Tamaño: (302, 167)
Clase: Supta Kapotasana, Imagen: File7.jpeg, Tamaño: (275, 183)
Clase: Salamba Sarvangasana, Imagen: File16.png, Tamaño: (258, 501)
Clase: Salamba Sarvangasana, Imagen: File17.png, Tamaño: (100, 275)
Clase: Salamba Sarvangasana, Imagen: File29.png, Tamaño: (213, 553)
Clase: Salamba Sarvangasana, Imagen: File28.png, Tamaño: (168, 422)
Clase: Salamba Sarvangasana, Imagen: File73.jpeg, Tamaño: (183, 275)
Clase: Ardha Navasana, Imagen: File4.jpeg, Tamaño: (255, 198)
Clase: Ardha Navasana, Imagen: File3.jpeg, Tamaño: (291, 173)
Clase: Ardha Navasana, Imagen: File12.png, Tamaño: (275, 183)
Clase: Ardha Navasana, Imagen: File11.jpeg, Tamaño: (294, 172)
Clase: Ardha Navasana, Imagen: File7.jpeg, Tamaño: (225, 225)
Clase: Virabhadrasana Two, Imagen: File16.png, Tamaño: (618, 559)
Clase: Virabhadrasana Two, Imagen: File17.png, Tamaño: (194, 170)
Clase: Virabhadrasana Two, Imagen: File29.png, Tamaño: (702, 540)
Clase: Virabhadrasana Two, Imagen: File28.png, Tamaño: (334, 270)
Clase: Virabhadrasana Two, Imagen: File45.png, Tamaño: (340, 372)
Clase: Upavistha Konasana, Imagen: File13.jpeg, Tamaño: (325, 155)
Clase: Upavistha Konasana, Imagen: File4.jpeg, Tamaño: (266, 190)
Clase: Upavistha Konasana, Imagen: File2.jpeg, Tamaño: (288, 175)
Clase: Upavistha Konasana, Imagen: File3.jpeg, Tamaño: (262, 192)
Clase: Upavistha Konasana, Imagen: File11.jpeg, Tamaño: (265, 190)
Clase: Pincha Mayurasana, Imagen: File16.png, Tamaño: (131, 500)
Clase: Pincha Mayurasana, Imagen: File17.png, Tamaño: (118, 337)

Clase: Pincha Mayurasana, Imagen: File29.png, Tamaño: (213, 491)
Clase: Pincha Mayurasana, Imagen: File28.png, Tamaño: (229, 517)
Clase: Pincha Mayurasana, Imagen: File42.jpeg, Tamaño: (201, 251)
Clase: Padmasana, Imagen: File16.png, Tamaño: (900, 675)
Clase: Padmasana, Imagen: File17.png, Tamaño: (602, 720)
Clase: Padmasana, Imagen: File29.png, Tamaño: (512, 512)
Clase: Padmasana, Imagen: File28.png, Tamaño: (363, 516)
Clase: Padmasana, Imagen: File73.jpeg, Tamaño: (275, 183)
Clase: Utthita Parsvakonasana, Imagen: File16.png, Tamaño: (531, 426)
Clase: Utthita Parsvakonasana, Imagen: File17.png, Tamaño: (653, 425)
Clase: Utthita Parsvakonasana, Imagen: File29.png, Tamaño: (750, 514)
Clase: Utthita Parsvakonasana, Imagen: File28.png, Tamaño: (447, 470)
Clase: Utthita Parsvakonasana, Imagen: File45.png, Tamaño: (765, 695)
Clase: Urdhva Mukha Svsnssana, Imagen: File16.png, Tamaño: (682, 472)
Clase: Urdhva Mukha Svsnssana, Imagen: File17.png, Tamaño: (328, 212)
Clase: Urdhva Mukha Svsnssana, Imagen: File29.png, Tamaño: (666, 474)
Clase: Urdhva Mukha Svsnssana, Imagen: File28.png, Tamaño: (422, 248)
Clase: Urdhva Mukha Svsnssana, Imagen: File45.png, Tamaño: (257, 186)
Clase: Balasana, Imagen: File16.png, Tamaño: (647, 528)
Clase: Balasana, Imagen: File45.jpg, Tamaño: (419, 230)
Clase: Balasana, Imagen: File17.png, Tamaño: (531, 208)
Clase: Balasana, Imagen: File29.png, Tamaño: (748, 344)
Clase: Balasana, Imagen: File71.png, Tamaño: (430, 221)
Clase: Bitilasana, Imagen: File16.png, Tamaño: (352, 256)
Clase: Bitilasana, Imagen: File17.png, Tamaño: (521, 392)
Clase: Bitilasana, Imagen: File29.png, Tamaño: (762, 578)
Clase: Bitilasana, Imagen: File71.png, Tamaño: (526, 417)
Clase: Bitilasana, Imagen: File28.png, Tamaño: (372, 294)
Clase: Trikonasana, Imagen: File13.jpeg, Tamaño: (225, 225)
Clase: Trikonasana, Imagen: File4.jpeg, Tamaño: (300, 168)
Clase: Trikonasana, Imagen: File2.jpeg, Tamaño: (208, 242)
Clase: Trikonasana, Imagen: File3.jpeg, Tamaño: (275, 183)
Clase: Trikonasana, Imagen: File11.jpeg, Tamaño: (275, 183)
Clase: Eka Pada Rajakapotasana, Imagen: File16.png, Tamaño: (278, 326)
Clase: Eka Pada Rajakapotasana, Imagen: File17.png, Tamaño: (498, 482)
Clase: Eka Pada Rajakapotasana, Imagen: File29.png, Tamaño: (562, 704)
Clase: Eka Pada Rajakapotasana, Imagen: File51.jpeg, Tamaño: (263, 192)
Clase: Eka Pada Rajakapotasana, Imagen: File28.png, Tamaño: (526, 583)
Clase: Navasana, Imagen: File13.jpeg, Tamaño: (275, 183)
Clase: Navasana, Imagen: File4.jpeg, Tamaño: (225, 225)
Clase: Navasana, Imagen: File2.jpeg, Tamaño: (283, 178)
Clase: Navasana, Imagen: File3.jpeg, Tamaño: (268, 188)
Clase: Navasana, Imagen: File11.jpeg, Tamaño: (279, 181)
Clase: Ardha Chandrasana, Imagen: File16.png, Tamaño: (404, 414)
Clase: Ardha Chandrasana, Imagen: File17.png, Tamaño: (513, 539)
Clase: Ardha Chandrasana, Imagen: File29.png, Tamaño: (446, 493)
Clase: Ardha Chandrasana, Imagen: File28.png, Tamaño: (397, 480)
Clase: Ardha Chandrasana, Imagen: File45.png, Tamaño: (584, 565)
Clase: Ashta Chandrasana, Imagen: File4.jpeg, Tamaño: (275, 183)
Clase: Ashta Chandrasana, Imagen: File2.jpeg, Tamaño: (105, 125)
Clase: Ashta Chandrasana, Imagen: File3.jpeg, Tamaño: (197, 256)
Clase: Ashta Chandrasana, Imagen: File12.png, Tamaño: (300, 168)
Clase: Ashta Chandrasana, Imagen: File11.jpeg, Tamaño: (224, 225)
Clase: Salamba Bhujangasana, Imagen: File16.png, Tamaño: (772, 292)
Clase: Salamba Bhujangasana, Imagen: File17.png, Tamaño: (433, 157)
Clase: Salamba Bhujangasana, Imagen: File29.png, Tamaño: (916, 473)

Clase: Salamba Bhujangasana, Imagen: File28.png, Tamaño: (497, 191)
Clase: Salamba Bhujangasana, Imagen: File45.png, Tamaño: (377, 246)
Clase: Adho Mukha Vrksasana, Imagen: File16.png, Tamaño: (137, 548)
Clase: Adho Mukha Vrksasana, Imagen: File17.png, Tamaño: (251, 744)
Clase: Adho Mukha Vrksasana, Imagen: File29.png, Tamaño: (119, 595)
Clase: Adho Mukha Vrksasana, Imagen: File28.png, Tamaño: (106, 593)
Clase: Adho Mukha Vrksasana, Imagen: File45.png, Tamaño: (168, 491)
Clase: Ardha Pincha Mayurasana, Imagen: File16.png, Tamaño: (359, 270)
Clase: Ardha Pincha Mayurasana, Imagen: File17.png, Tamaño: (471, 453)
Clase: Ardha Pincha Mayurasana, Imagen: File29.png, Tamaño: (184, 133)
Clase: Ardha Pincha Mayurasana, Imagen: File51.jpeg, Tamaño: (309, 163)
Clase: Ardha Pincha Mayurasana, Imagen: File28.png, Tamaño: (709, 431)
Clase: Dhanurasana, Imagen: File16.png, Tamaño: (350, 311)
Clase: Dhanurasana, Imagen: File17.png, Tamaño: (448, 279)
Clase: Dhanurasana, Imagen: File29.png, Tamaño: (425, 401)
Clase: Dhanurasana, Imagen: File51.jpeg, Tamaño: (290, 174)
Clase: Dhanurasana, Imagen: File28.png, Tamaño: (326, 269)
Clase: Virabhadrasana One, Imagen: File16.png, Tamaño: (515, 714)
Clase: Virabhadrasana One, Imagen: File17.png, Tamaño: (191, 296)
Clase: Virabhadrasana One, Imagen: File29.png, Tamaño: (226, 260)
Clase: Virabhadrasana One, Imagen: File28.png, Tamaño: (219, 274)
Clase: Virabhadrasana One, Imagen: File45.png, Tamaño: (279, 466)
Clase: Utkatasana, Imagen: File16.png, Tamaño: (234, 322)
Clase: Utkatasana, Imagen: File17.png, Tamaño: (357, 468)
Clase: Utkatasana, Imagen: File29.png, Tamaño: (187, 368)
Clase: Utkatasana, Imagen: File71.png, Tamaño: (187, 400)
Clase: Utkatasana, Imagen: File28.png, Tamaño: (155, 225)
Clase: Urdhva Dhanurasana, Imagen: File16.png, Tamaño: (616, 357)
Clase: Urdhva Dhanurasana, Imagen: File17.png, Tamaño: (715, 460)
Clase: Urdhva Dhanurasana, Imagen: File29.png, Tamaño: (141, 139)
Clase: Urdhva Dhanurasana, Imagen: File28.png, Tamaño: (595, 472)
Clase: Urdhva Dhanurasana, Imagen: File73.jpeg, Tamaño: (310, 163)
Clase: Halasana, Imagen: File16.png, Tamaño: (463, 236)
Clase: Halasana, Imagen: File17.png, Tamaño: (565, 253)
Clase: Halasana, Imagen: File29.png, Tamaño: (360, 159)
Clase: Halasana, Imagen: File28.png, Tamaño: (581, 214)
Clase: Halasana, Imagen: File45.png, Tamaño: (284, 235)
Clase: Ustrasana, Imagen: File16.png, Tamaño: (253, 299)
Clase: Ustrasana, Imagen: File17.png, Tamaño: (348, 415)
Clase: Ustrasana, Imagen: File29.png, Tamaño: (270, 340)
Clase: Ustrasana, Imagen: File71.png, Tamaño: (204, 330)
Clase: Ustrasana, Imagen: File28.png, Tamaño: (282, 322)
Clase: Alanasana, Imagen: File4.jpeg, Tamaño: (275, 183)
Clase: Alanasana, Imagen: File3.jpeg, Tamaño: (188, 268)
Clase: Alanasana, Imagen: File7.jpeg, Tamaño: (299, 168)
Clase: Alanasana, Imagen: File5.png, Tamaño: (209, 241)
Clase: Alanasana, Imagen: File15.jpeg, Tamaño: (225, 225)
Clase: Parsvottanasana, Imagen: File16.png, Tamaño: (366, 268)
Clase: Parsvottanasana, Imagen: File17.png, Tamaño: (567, 436)
Clase: Parsvottanasana, Imagen: File29.png, Tamaño: (181, 277)
Clase: Parsvottanasana, Imagen: File28.png, Tamaño: (560, 372)
Clase: Parsvottanasana, Imagen: File42.jpeg, Tamaño: (274, 184)
Clase: Vrksasana, Imagen: File16.png, Tamaño: (253, 851)
Clase: Vrksasana, Imagen: File17.png, Tamaño: (240, 702)
Clase: Vrksasana, Imagen: File29.png, Tamaño: (92, 307)
Clase: Vrksasana, Imagen: File28.png, Tamaño: (149, 333)

```

Clase: Vrksasana, Imagen: File45.png, Tamaño: (198, 393)
Clase: Garudasana, Imagen: File16.png, Tamaño: (135, 489)
Clase: Garudasana, Imagen: File17.png, Tamaño: (100, 475)
Clase: Garudasana, Imagen: File29.png, Tamaño: (117, 530)
Clase: Garudasana, Imagen: File71.png, Tamaño: (137, 489)
Clase: Garudasana, Imagen: File28.png, Tamaño: (207, 543)
Clase: Paschimottanasana, Imagen: File16.png, Tamaño: (254, 204)
Clase: Paschimottanasana, Imagen: File17.png, Tamaño: (646, 491)
Clase: Paschimottanasana, Imagen: File29.png, Tamaño: (354, 289)
Clase: Paschimottanasana, Imagen: File28.png, Tamaño: (533, 216)
Clase: Paschimottanasana, Imagen: File45.png, Tamaño: (377, 167)

```

Como podemos apreciar, las 5 primeras imágenes de cada clase no son consistentes en términos de tamaño. Por ello, es interesante redimensionar las imágenes para que no afecte a los modelos que se van a emplear posteriormente.

Verificar si hay imágenes corruptas

```

In [ ]: def verificar_imagenes_corruptas(base_path):
        for clase in os.listdir(base_path):
            ruta_clase = os.path.join(base_path, clase)
            if os.path.isdir(ruta_clase):
                imagenes = [f for f in os.listdir(ruta_clase) if f.lower().endsw
                for img_file in imagenes:
                    img_path = os.path.join(ruta_clase, img_file)
                    try:
                        img = Image.open(img_path)
                        img.verify() # Verifica si la imagen está corrupta
                    except (IOError, SyntaxError) as e:
                        print(f"Imagen corrupta: {img_path}")

verificar_imagenes_corruptas(folder_path)

```

Como podemos ver, no hay ninguna imagen corrupta

Analizar el formato y tipo de imagen

```

In [ ]: def verificar_tipo_imagen(base_path):
        for clase in os.listdir(base_path):
            ruta_clase = os.path.join(base_path, clase)
            if os.path.isdir(ruta_clase):
                imagenes = [f for f in os.listdir(ruta_clase) if f.lower().endsw
                for img_file in imagenes:
                    img_path = os.path.join(ruta_clase, img_file)
                    with Image.open(img_path) as img:
                        print(f"Clase: {clase}, Imagen: {img_file}, Formato: {im

verificar_tipo_imagen(folder_path)

```

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

Clase: Phalakasana, Imagen: File58.jpeg, Formato: JPEG
Clase: Phalakasana, Imagen: File12.png, Formato: PNG
Clase: Phalakasana, Imagen: File32.png, Formato: PNG
Clase: Phalakasana, Imagen: File52.jpg, Formato: JPEG
Clase: Phalakasana, Imagen: File64.jpeg, Formato: JPEG
Clase: Phalakasana, Imagen: File59.jpeg, Formato: JPEG
Clase: Phalakasana, Imagen: File65.jpeg, Formato: JPEG
Clase: Phalakasana, Imagen: File49.png, Formato: PNG
Clase: Phalakasana, Imagen: File5.png, Formato: PNG
Clase: Phalakasana, Imagen: File57.jpg, Formato: JPEG
Clase: Phalakasana, Imagen: File37.png, Formato: PNG
Clase: Phalakasana, Imagen: File44.png, Formato: PNG
Clase: Phalakasana, Imagen: File19.png, Formato: PNG
Clase: Phalakasana, Imagen: File8.png, Formato: PNG
Clase: Phalakasana, Imagen: File63.jpeg, Formato: JPEG
Clase: Phalakasana, Imagen: File1.png, Formato: PNG
Clase: Phalakasana, Imagen: File33.png, Formato: PNG
Clase: Phalakasana, Imagen: File66.jpeg, Formato: JPEG
Clase: Phalakasana, Imagen: File53.jpg, Formato: JPEG
Clase: Phalakasana, Imagen: File20.png, Formato: PNG
Clase: Phalakasana, Imagen: File23.png, Formato: PNG
Clase: Phalakasana, Imagen: File46.png, Formato: PNG
Clase: Phalakasana, Imagen: File41.png, Formato: PNG
Clase: Phalakasana, Imagen: File55.jpg, Formato: JPEG
Clase: Phalakasana, Imagen: File40.png, Formato: PNG
Clase: Phalakasana, Imagen: File9.png, Formato: PNG
Clase: Phalakasana, Imagen: File39.png, Formato: PNG
Clase: Phalakasana, Imagen: File48.png, Formato: PNG
Clase: Phalakasana, Imagen: File60.jpeg, Formato: JPEG
Clase: Phalakasana, Imagen: File34.png, Formato: PNG
Clase: Phalakasana, Imagen: File13.png, Formato: PNG
Clase: Phalakasana, Imagen: File10.png, Formato: PNG
Clase: Phalakasana, Imagen: File11.png, Formato: PNG
Clase: Phalakasana, Imagen: File54.jpg, Formato: JPEG
Clase: Phalakasana, Imagen: File25.png, Formato: PNG
Clase: Phalakasana, Imagen: File38.png, Formato: PNG
Clase: Phalakasana, Imagen: File31.png, Formato: PNG
Clase: Phalakasana, Imagen: File35.png, Formato: PNG
Clase: Phalakasana, Imagen: File2.png, Formato: PNG
Clase: Phalakasana, Imagen: File51.jpg, Formato: JPEG
Clase: Phalakasana, Imagen: File36.png, Formato: PNG
Clase: Phalakasana, Imagen: File7.png, Formato: PNG
Clase: Phalakasana, Imagen: File47.png, Formato: PNG
Clase: Phalakasana, Imagen: File14.png, Formato: PNG
Clase: Phalakasana, Imagen: File6.png, Formato: PNG
Clase: Phalakasana, Imagen: File18.png, Formato: PNG
Clase: Phalakasana, Imagen: File27.png, Formato: PNG
Clase: Phalakasana, Imagen: File22.png, Formato: PNG
Clase: Phalakasana, Imagen: File42.png, Formato: PNG
Clase: Phalakasana, Imagen: File43.png, Formato: PNG
Clase: Phalakasana, Imagen: File26.png, Formato: PNG
Clase: Phalakasana, Imagen: File56.jpg, Formato: JPEG
Clase: Virabhadrasana Three, Imagen: File16.png, Formato: PNG
Clase: Virabhadrasana Three, Imagen: File17.png, Formato: PNG
Clase: Virabhadrasana Three, Imagen: File29.png, Formato: PNG
Clase: Virabhadrasana Three, Imagen: File28.png, Formato: PNG

[illegible]

[illegible]

Clase: Camatkarasana, Imagen: File56.jpeg, Formato: JPEG
Clase: Camatkarasana, Imagen: File55.jpeg, Formato: JPEG
Clase: Camatkarasana, Imagen: File51.png, Formato: PNG
Clase: Camatkarasana, Imagen: File36.png, Formato: PNG
Clase: Camatkarasana, Imagen: File7.png, Formato: PNG
Clase: Camatkarasana, Imagen: File47.png, Formato: PNG
Clase: Camatkarasana, Imagen: File14.png, Formato: PNG
Clase: Camatkarasana, Imagen: File6.png, Formato: PNG
Clase: Camatkarasana, Imagen: File18.png, Formato: PNG
Clase: Camatkarasana, Imagen: File27.png, Formato: PNG
Clase: Camatkarasana, Imagen: File22.png, Formato: PNG
Clase: Camatkarasana, Imagen: File42.png, Formato: PNG
Clase: Camatkarasana, Imagen: File43.png, Formato: PNG
Clase: Camatkarasana, Imagen: File50.png, Formato: PNG
Clase: Camatkarasana, Imagen: File26.png, Formato: PNG
Clase: Parsva Virabhadrasana, Imagen: File13.jpeg, Formato: JPEG
Clase: Parsva Virabhadrasana, Imagen: File4.jpeg, Formato: JPEG
Clase: Parsva Virabhadrasana, Imagen: File2.jpeg, Formato: JPEG
Clase: Parsva Virabhadrasana, Imagen: File3.jpeg, Formato: JPEG
Clase: Parsva Virabhadrasana, Imagen: File11.jpeg, Formato: JPEG
Clase: Parsva Virabhadrasana, Imagen: File7.jpeg, Formato: JPEG
Clase: Parsva Virabhadrasana, Imagen: File9.jpeg, Formato: JPEG
Clase: Parsva Virabhadrasana, Imagen: File14.jpeg, Formato: JPEG
Clase: Parsva Virabhadrasana, Imagen: File10.jpeg, Formato: JPEG
Clase: Parsva Virabhadrasana, Imagen: File8.jpeg, Formato: JPEG
Clase: Parsva Virabhadrasana, Imagen: File12.jpeg, Formato: JPEG
Clase: Parsva Virabhadrasana, Imagen: File1.jpeg, Formato: JPEG
Clase: Parsva Virabhadrasana, Imagen: File5.jpeg, Formato: JPEG
Clase: Parsva Virabhadrasana, Imagen: File6.jpeg, Formato: JPEG
Clase: Hanumanasana, Imagen: File16.png, Formato: PNG
Clase: Hanumanasana, Imagen: File17.png, Formato: PNG
Clase: Hanumanasana, Imagen: File29.png, Formato: PNG
Clase: Hanumanasana, Imagen: File28.png, Formato: PNG
Clase: Hanumanasana, Imagen: File15.png, Formato: PNG
Clase: Hanumanasana, Imagen: File30.png, Formato: PNG
Clase: Hanumanasana, Imagen: File3.png, Formato: PNG
Clase: Hanumanasana, Imagen: File21.png, Formato: PNG
Clase: Hanumanasana, Imagen: File4.png, Formato: PNG
Clase: Hanumanasana, Imagen: File24.png, Formato: PNG
Clase: Hanumanasana, Imagen: File36.jpeg, Formato: JPEG
Clase: Hanumanasana, Imagen: File12.png, Formato: PNG
Clase: Hanumanasana, Imagen: File32.png, Formato: PNG
Clase: Hanumanasana, Imagen: File39.jpeg, Formato: JPEG
Clase: Hanumanasana, Imagen: File5.png, Formato: PNG
Clase: Hanumanasana, Imagen: File41.jpeg, Formato: JPEG
Clase: Hanumanasana, Imagen: File38.jpeg, Formato: JPEG
Clase: Hanumanasana, Imagen: File37.jpeg, Formato: JPEG
Clase: Hanumanasana, Imagen: File19.png, Formato: PNG
Clase: Hanumanasana, Imagen: File8.png, Formato: PNG
Clase: Hanumanasana, Imagen: File1.png, Formato: PNG
Clase: Hanumanasana, Imagen: File33.png, Formato: PNG
Clase: Hanumanasana, Imagen: File20.png, Formato: PNG
Clase: Hanumanasana, Imagen: File23.png, Formato: PNG
Clase: Hanumanasana, Imagen: File9.png, Formato: PNG
Clase: Hanumanasana, Imagen: File34.png, Formato: PNG
Clase: Hanumanasana, Imagen: File13.png, Formato: PNG

[illegible]

Clase: Setu Bandha Sarvangasana, Imagen: File9.png, Formato: PNG
Clase: Setu Bandha Sarvangasana, Imagen: File39.png, Formato: PNG
Clase: Setu Bandha Sarvangasana, Imagen: File13.png, Formato: PNG
Clase: Setu Bandha Sarvangasana, Imagen: File10.png, Formato: PNG
Clase: Setu Bandha Sarvangasana, Imagen: File11.png, Formato: PNG
Clase: Setu Bandha Sarvangasana, Imagen: File54.jpg, Formato: JPEG
Clase: Setu Bandha Sarvangasana, Imagen: File25.png, Formato: PNG
Clase: Setu Bandha Sarvangasana, Imagen: File38.png, Formato: PNG
Clase: Setu Bandha Sarvangasana, Imagen: File31.png, Formato: PNG
Clase: Setu Bandha Sarvangasana, Imagen: File35.png, Formato: PNG
Clase: Setu Bandha Sarvangasana, Imagen: File2.png, Formato: PNG
Clase: Setu Bandha Sarvangasana, Imagen: File69.png, Formato: PNG
Clase: Setu Bandha Sarvangasana, Imagen: File36.png, Formato: PNG
Clase: Setu Bandha Sarvangasana, Imagen: File61.jpg, Formato: JPEG
Clase: Setu Bandha Sarvangasana, Imagen: File7.png, Formato: PNG
Clase: Setu Bandha Sarvangasana, Imagen: File47.png, Formato: PNG
Clase: Setu Bandha Sarvangasana, Imagen: File14.png, Formato: PNG
Clase: Setu Bandha Sarvangasana, Imagen: File6.png, Formato: PNG
Clase: Setu Bandha Sarvangasana, Imagen: File18.png, Formato: PNG
Clase: Setu Bandha Sarvangasana, Imagen: File27.png, Formato: PNG
Clase: Setu Bandha Sarvangasana, Imagen: File22.png, Formato: PNG
Clase: Setu Bandha Sarvangasana, Imagen: File42.png, Formato: PNG
Clase: Setu Bandha Sarvangasana, Imagen: File43.png, Formato: PNG
Clase: Setu Bandha Sarvangasana, Imagen: File26.png, Formato: PNG
Clase: Malasana, Imagen: File16.png, Formato: PNG
Clase: Malasana, Imagen: File17.png, Formato: PNG
Clase: Malasana, Imagen: File29.png, Formato: PNG
Clase: Malasana, Imagen: File28.png, Formato: PNG
Clase: Malasana, Imagen: File73.jpeg, Formato: JPEG
Clase: Malasana, Imagen: File45.png, Formato: PNG
Clase: Malasana, Imagen: File56.png, Formato: PNG
Clase: Malasana, Imagen: File15.png, Formato: PNG
Clase: Malasana, Imagen: File30.png, Formato: PNG
Clase: Malasana, Imagen: File3.png, Formato: PNG
Clase: Malasana, Imagen: File21.png, Formato: PNG
Clase: Malasana, Imagen: File4.png, Formato: PNG
Clase: Malasana, Imagen: File71.jpeg, Formato: JPEG
Clase: Malasana, Imagen: File24.png, Formato: PNG
Clase: Malasana, Imagen: File53.png, Formato: PNG
Clase: Malasana, Imagen: File64.png, Formato: PNG
Clase: Malasana, Imagen: File72.jpeg, Formato: JPEG
Clase: Malasana, Imagen: File12.png, Formato: PNG
Clase: Malasana, Imagen: File32.png, Formato: PNG
Clase: Malasana, Imagen: File52.png, Formato: PNG
Clase: Malasana, Imagen: File54.png, Formato: PNG
Clase: Malasana, Imagen: File55.png, Formato: PNG
Clase: Malasana, Imagen: File49.png, Formato: PNG
Clase: Malasana, Imagen: File5.png, Formato: PNG
Clase: Malasana, Imagen: File65.png, Formato: PNG
Clase: Malasana, Imagen: File69.jpeg, Formato: JPEG
Clase: Malasana, Imagen: File59.png, Formato: PNG
Clase: Malasana, Imagen: File37.png, Formato: PNG
Clase: Malasana, Imagen: File44.png, Formato: PNG
Clase: Malasana, Imagen: File19.png, Formato: PNG
Clase: Malasana, Imagen: File61.png, Formato: PNG
Clase: Malasana, Imagen: File8.png, Formato: PNG

Clase: Malasana, Imagen: File60.png, Formato: PNG
Clase: Malasana, Imagen: File1.png, Formato: PNG
Clase: Malasana, Imagen: File33.png, Formato: PNG
Clase: Malasana, Imagen: File20.png, Formato: PNG
Clase: Malasana, Imagen: File23.png, Formato: PNG
Clase: Malasana, Imagen: File46.png, Formato: PNG
Clase: Malasana, Imagen: File41.png, Formato: PNG
Clase: Malasana, Imagen: File40.png, Formato: PNG
Clase: Malasana, Imagen: File70.jpeg, Formato: JPEG
Clase: Malasana, Imagen: File9.png, Formato: PNG
Clase: Malasana, Imagen: File39.png, Formato: PNG
Clase: Malasana, Imagen: File48.png, Formato: PNG
Clase: Malasana, Imagen: File34.png, Formato: PNG
Clase: Malasana, Imagen: File13.png, Formato: PNG
Clase: Malasana, Imagen: File10.png, Formato: PNG
Clase: Malasana, Imagen: File66.png, Formato: PNG
Clase: Malasana, Imagen: File67.png, Formato: PNG
Clase: Malasana, Imagen: File11.png, Formato: PNG
Clase: Malasana, Imagen: File25.png, Formato: PNG
Clase: Malasana, Imagen: File38.png, Formato: PNG
Clase: Malasana, Imagen: File31.png, Formato: PNG
Clase: Malasana, Imagen: File35.png, Formato: PNG
Clase: Malasana, Imagen: File2.png, Formato: PNG
Clase: Malasana, Imagen: File57.png, Formato: PNG
Clase: Malasana, Imagen: File68.png, Formato: PNG
Clase: Malasana, Imagen: File51.png, Formato: PNG
Clase: Malasana, Imagen: File36.png, Formato: PNG
Clase: Malasana, Imagen: File7.png, Formato: PNG
Clase: Malasana, Imagen: File47.png, Formato: PNG
Clase: Malasana, Imagen: File14.png, Formato: PNG
Clase: Malasana, Imagen: File6.png, Formato: PNG
Clase: Malasana, Imagen: File18.png, Formato: PNG
Clase: Malasana, Imagen: File27.png, Formato: PNG
Clase: Malasana, Imagen: File22.png, Formato: PNG
Clase: Malasana, Imagen: File42.png, Formato: PNG
Clase: Malasana, Imagen: File63.png, Formato: PNG
Clase: Malasana, Imagen: File62.png, Formato: PNG
Clase: Malasana, Imagen: File43.png, Formato: PNG
Clase: Malasana, Imagen: File50.png, Formato: PNG
Clase: Malasana, Imagen: File58.png, Formato: PNG
Clase: Malasana, Imagen: File26.png, Formato: PNG
Clase: Sivasana, Imagen: File13.jpeg, Formato: JPEG
Clase: Sivasana, Imagen: File4.jpeg, Formato: JPEG
Clase: Sivasana, Imagen: File3.jpeg, Formato: JPEG
Clase: Sivasana, Imagen: File11.jpeg, Formato: JPEG
Clase: Sivasana, Imagen: File20.jpeg, Formato: JPEG
Clase: Sivasana, Imagen: File7.jpeg, Formato: JPEG
Clase: Sivasana, Imagen: File18.jpeg, Formato: JPEG
Clase: Sivasana, Imagen: File15.jpeg, Formato: JPEG
Clase: Sivasana, Imagen: File8.png, Formato: PNG
Clase: Sivasana, Imagen: File9.jpeg, Formato: JPEG
Clase: Sivasana, Imagen: File16.jpeg, Formato: JPEG
Clase: Sivasana, Imagen: File14.jpeg, Formato: JPEG
Clase: Sivasana, Imagen: File10.jpeg, Formato: JPEG
Clase: Sivasana, Imagen: File2.png, Formato: PNG
Clase: Sivasana, Imagen: File17.jpeg, Formato: JPEG

[illegible]

Clase: Utthita Hasta Padangusthasana, Imagen: File36.png, Formato: PNG
Clase: Utthita Hasta Padangusthasana, Imagen: File7.png, Formato: PNG
Clase: Utthita Hasta Padangusthasana, Imagen: File47.png, Formato: PNG
Clase: Utthita Hasta Padangusthasana, Imagen: File14.png, Formato: PNG
Clase: Utthita Hasta Padangusthasana, Imagen: File6.png, Formato: PNG
Clase: Utthita Hasta Padangusthasana, Imagen: File18.png, Formato: PNG
Clase: Utthita Hasta Padangusthasana, Imagen: File27.png, Formato: PNG
Clase: Utthita Hasta Padangusthasana, Imagen: File22.png, Formato: PNG
Clase: Utthita Hasta Padangusthasana, Imagen: File42.png, Formato: PNG
Clase: Utthita Hasta Padangusthasana, Imagen: File43.png, Formato: PNG
Clase: Utthita Hasta Padangusthasana, Imagen: File50.png, Formato: PNG
Clase: Utthita Hasta Padangusthasana, Imagen: File58.png, Formato: PNG
Clase: Utthita Hasta Padangusthasana, Imagen: File26.png, Formato: PNG
Clase: Bakasana, Imagen: File16.png, Formato: PNG
Clase: Bakasana, Imagen: File17.png, Formato: PNG
Clase: Bakasana, Imagen: File29.png, Formato: PNG
Clase: Bakasana, Imagen: File71.png, Formato: PNG
Clase: Bakasana, Imagen: File28.png, Formato: PNG
Clase: Bakasana, Imagen: File45.png, Formato: PNG
Clase: Bakasana, Imagen: File56.png, Formato: PNG
Clase: Bakasana, Imagen: File80.jpeg, Formato: JPEG
Clase: Bakasana, Imagen: File15.png, Formato: PNG
Clase: Bakasana, Imagen: File79.jpeg, Formato: JPEG
Clase: Bakasana, Imagen: File30.png, Formato: PNG
Clase: Bakasana, Imagen: File3.png, Formato: PNG
Clase: Bakasana, Imagen: File75.png, Formato: PNG
Clase: Bakasana, Imagen: File21.png, Formato: PNG
Clase: Bakasana, Imagen: File4.png, Formato: PNG
Clase: Bakasana, Imagen: File74.png, Formato: PNG
Clase: Bakasana, Imagen: File24.png, Formato: PNG
Clase: Bakasana, Imagen: File53.png, Formato: PNG
Clase: Bakasana, Imagen: File81.jpeg, Formato: JPEG
Clase: Bakasana, Imagen: File70.png, Formato: PNG
Clase: Bakasana, Imagen: File72.png, Formato: PNG
Clase: Bakasana, Imagen: File64.png, Formato: PNG
Clase: Bakasana, Imagen: File12.png, Formato: PNG
Clase: Bakasana, Imagen: File32.png, Formato: PNG
Clase: Bakasana, Imagen: File52.png, Formato: PNG
Clase: Bakasana, Imagen: File54.png, Formato: PNG
Clase: Bakasana, Imagen: File55.png, Formato: PNG
Clase: Bakasana, Imagen: File49.png, Formato: PNG
Clase: Bakasana, Imagen: File5.png, Formato: PNG
Clase: Bakasana, Imagen: File65.png, Formato: PNG
Clase: Bakasana, Imagen: File59.png, Formato: PNG
Clase: Bakasana, Imagen: File37.png, Formato: PNG
Clase: Bakasana, Imagen: File44.png, Formato: PNG
Clase: Bakasana, Imagen: File19.png, Formato: PNG
Clase: Bakasana, Imagen: File61.png, Formato: PNG
Clase: Bakasana, Imagen: File8.png, Formato: PNG
Clase: Bakasana, Imagen: File60.png, Formato: PNG
Clase: Bakasana, Imagen: File1.png, Formato: PNG
Clase: Bakasana, Imagen: File33.png, Formato: PNG
Clase: Bakasana, Imagen: File20.png, Formato: PNG
Clase: Bakasana, Imagen: File23.png, Formato: PNG
Clase: Bakasana, Imagen: File46.png, Formato: PNG
Clase: Bakasana, Imagen: File41.png, Formato: PNG

Clase: Bakasana, Imagen: File40.png, Formato: PNG
Clase: Bakasana, Imagen: File9.png, Formato: PNG
Clase: Bakasana, Imagen: File39.png, Formato: PNG
Clase: Bakasana, Imagen: File48.png, Formato: PNG
Clase: Bakasana, Imagen: File84.png, Formato: PNG
Clase: Bakasana, Imagen: File34.png, Formato: PNG
Clase: Bakasana, Imagen: File13.png, Formato: PNG
Clase: Bakasana, Imagen: File10.png, Formato: PNG
Clase: Bakasana, Imagen: File66.png, Formato: PNG
Clase: Bakasana, Imagen: File67.png, Formato: PNG
Clase: Bakasana, Imagen: File11.png, Formato: PNG
Clase: Bakasana, Imagen: File25.png, Formato: PNG
Clase: Bakasana, Imagen: File38.png, Formato: PNG
Clase: Bakasana, Imagen: File83.jpeg, Formato: JPEG
Clase: Bakasana, Imagen: File31.png, Formato: PNG
Clase: Bakasana, Imagen: File35.png, Formato: PNG
Clase: Bakasana, Imagen: File2.png, Formato: PNG
Clase: Bakasana, Imagen: File77.png, Formato: PNG
Clase: Bakasana, Imagen: File57.png, Formato: PNG
Clase: Bakasana, Imagen: File68.png, Formato: PNG
Clase: Bakasana, Imagen: File51.png, Formato: PNG
Clase: Bakasana, Imagen: File69.png, Formato: PNG
Clase: Bakasana, Imagen: File36.png, Formato: PNG
Clase: Bakasana, Imagen: File82.jpeg, Formato: JPEG
Clase: Bakasana, Imagen: File7.png, Formato: PNG
Clase: Bakasana, Imagen: File47.png, Formato: PNG
Clase: Bakasana, Imagen: File76.png, Formato: PNG
Clase: Bakasana, Imagen: File14.png, Formato: PNG
Clase: Bakasana, Imagen: File6.png, Formato: PNG
Clase: Bakasana, Imagen: File18.png, Formato: PNG
Clase: Bakasana, Imagen: File27.png, Formato: PNG
Clase: Bakasana, Imagen: File22.png, Formato: PNG
Clase: Bakasana, Imagen: File78.jpeg, Formato: JPEG
Clase: Bakasana, Imagen: File42.png, Formato: PNG
Clase: Bakasana, Imagen: File63.png, Formato: PNG
Clase: Bakasana, Imagen: File73.png, Formato: PNG
Clase: Bakasana, Imagen: File62.png, Formato: PNG
Clase: Bakasana, Imagen: File43.png, Formato: PNG
Clase: Bakasana, Imagen: File50.png, Formato: PNG
Clase: Bakasana, Imagen: File58.png, Formato: PNG
Clase: Bakasana, Imagen: File26.png, Formato: PNG
Clase: Vasisthasana, Imagen: File16.png, Formato: PNG
Clase: Vasisthasana, Imagen: File17.png, Formato: PNG
Clase: Vasisthasana, Imagen: File29.png, Formato: PNG
Clase: Vasisthasana, Imagen: File71.png, Formato: PNG
Clase: Vasisthasana, Imagen: File28.png, Formato: PNG
Clase: Vasisthasana, Imagen: File45.png, Formato: PNG
Clase: Vasisthasana, Imagen: File56.png, Formato: PNG
Clase: Vasisthasana, Imagen: File80.jpeg, Formato: JPEG
Clase: Vasisthasana, Imagen: File15.png, Formato: PNG
Clase: Vasisthasana, Imagen: File79.jpeg, Formato: JPEG
Clase: Vasisthasana, Imagen: File30.png, Formato: PNG
Clase: Vasisthasana, Imagen: File3.png, Formato: PNG
Clase: Vasisthasana, Imagen: File21.png, Formato: PNG
Clase: Vasisthasana, Imagen: File4.png, Formato: PNG
Clase: Vasisthasana, Imagen: File74.png, Formato: PNG

[illegible]

Clase: Vasisthasana, Imagen: File42.png, Formato: PNG
Clase: Vasisthasana, Imagen: File63.png, Formato: PNG
Clase: Vasisthasana, Imagen: File73.png, Formato: PNG
Clase: Vasisthasana, Imagen: File76.jpeg, Formato: JPEG
Clase: Vasisthasana, Imagen: File62.png, Formato: PNG
Clase: Vasisthasana, Imagen: File43.png, Formato: PNG
Clase: Vasisthasana, Imagen: File50.png, Formato: PNG
Clase: Vasisthasana, Imagen: File58.png, Formato: PNG
Clase: Vasisthasana, Imagen: File26.png, Formato: PNG
Clase: Anjaneyasana, Imagen: File16.png, Formato: PNG
Clase: Anjaneyasana, Imagen: File17.png, Formato: PNG
Clase: Anjaneyasana, Imagen: File29.png, Formato: PNG
Clase: Anjaneyasana, Imagen: File28.png, Formato: PNG
Clase: Anjaneyasana, Imagen: File45.png, Formato: PNG
Clase: Anjaneyasana, Imagen: File56.png, Formato: PNG
Clase: Anjaneyasana, Imagen: File15.png, Formato: PNG
Clase: Anjaneyasana, Imagen: File30.png, Formato: PNG
Clase: Anjaneyasana, Imagen: File3.png, Formato: PNG
Clase: Anjaneyasana, Imagen: File21.png, Formato: PNG
Clase: Anjaneyasana, Imagen: File4.png, Formato: PNG
Clase: Anjaneyasana, Imagen: File71.jpeg, Formato: JPEG
Clase: Anjaneyasana, Imagen: File24.png, Formato: PNG
Clase: Anjaneyasana, Imagen: File67.jpeg, Formato: JPEG
Clase: Anjaneyasana, Imagen: File53.png, Formato: PNG
Clase: Anjaneyasana, Imagen: File64.png, Formato: PNG
Clase: Anjaneyasana, Imagen: File12.png, Formato: PNG
Clase: Anjaneyasana, Imagen: File32.png, Formato: PNG
Clase: Anjaneyasana, Imagen: File52.png, Formato: PNG
Clase: Anjaneyasana, Imagen: File54.png, Formato: PNG
Clase: Anjaneyasana, Imagen: File55.png, Formato: PNG
Clase: Anjaneyasana, Imagen: File65.jpeg, Formato: JPEG
Clase: Anjaneyasana, Imagen: File49.png, Formato: PNG
Clase: Anjaneyasana, Imagen: File5.png, Formato: PNG
Clase: Anjaneyasana, Imagen: File69.jpeg, Formato: JPEG
Clase: Anjaneyasana, Imagen: File18.jpeg, Formato: JPEG
Clase: Anjaneyasana, Imagen: File59.png, Formato: PNG
Clase: Anjaneyasana, Imagen: File37.png, Formato: PNG
Clase: Anjaneyasana, Imagen: File44.png, Formato: PNG
Clase: Anjaneyasana, Imagen: File19.png, Formato: PNG
Clase: Anjaneyasana, Imagen: File61.png, Formato: PNG
Clase: Anjaneyasana, Imagen: File8.png, Formato: PNG
Clase: Anjaneyasana, Imagen: File60.png, Formato: PNG
Clase: Anjaneyasana, Imagen: File1.png, Formato: PNG
Clase: Anjaneyasana, Imagen: File33.png, Formato: PNG
Clase: Anjaneyasana, Imagen: File66.jpeg, Formato: JPEG
Clase: Anjaneyasana, Imagen: File20.png, Formato: PNG
Clase: Anjaneyasana, Imagen: File23.png, Formato: PNG
Clase: Anjaneyasana, Imagen: File46.png, Formato: PNG
Clase: Anjaneyasana, Imagen: File41.png, Formato: PNG
Clase: Anjaneyasana, Imagen: File40.png, Formato: PNG
Clase: Anjaneyasana, Imagen: File70.jpeg, Formato: JPEG
Clase: Anjaneyasana, Imagen: File68.jpeg, Formato: JPEG
Clase: Anjaneyasana, Imagen: File9.png, Formato: PNG
Clase: Anjaneyasana, Imagen: File39.png, Formato: PNG
Clase: Anjaneyasana, Imagen: File48.png, Formato: PNG
Clase: Anjaneyasana, Imagen: File34.png, Formato: PNG

[illegible]

[illegible]

[illegible]

[illegible]

Clase: Pincha Mayurasana, Imagen: File23.png, Formato: PNG
Clase: Pincha Mayurasana, Imagen: File43.jpeg, Formato: JPEG
Clase: Pincha Mayurasana, Imagen: File9.png, Formato: PNG
Clase: Pincha Mayurasana, Imagen: File34.png, Formato: PNG
Clase: Pincha Mayurasana, Imagen: File13.png, Formato: PNG
Clase: Pincha Mayurasana, Imagen: File10.png, Formato: PNG
Clase: Pincha Mayurasana, Imagen: File11.png, Formato: PNG
Clase: Pincha Mayurasana, Imagen: File25.png, Formato: PNG
Clase: Pincha Mayurasana, Imagen: File40.jpeg, Formato: JPEG
Clase: Pincha Mayurasana, Imagen: File31.png, Formato: PNG
Clase: Pincha Mayurasana, Imagen: File35.png, Formato: PNG
Clase: Pincha Mayurasana, Imagen: File2.png, Formato: PNG
Clase: Pincha Mayurasana, Imagen: File7.png, Formato: PNG
Clase: Pincha Mayurasana, Imagen: File14.png, Formato: PNG
Clase: Pincha Mayurasana, Imagen: File6.png, Formato: PNG
Clase: Pincha Mayurasana, Imagen: File18.png, Formato: PNG
Clase: Pincha Mayurasana, Imagen: File27.png, Formato: PNG
Clase: Pincha Mayurasana, Imagen: File22.png, Formato: PNG
Clase: Pincha Mayurasana, Imagen: File26.png, Formato: PNG
Clase: Padmasana, Imagen: File16.png, Formato: PNG
Clase: Padmasana, Imagen: File17.png, Formato: PNG
Clase: Padmasana, Imagen: File29.png, Formato: PNG
Clase: Padmasana, Imagen: File28.png, Formato: PNG
Clase: Padmasana, Imagen: File73.jpeg, Formato: JPEG
Clase: Padmasana, Imagen: File45.png, Formato: PNG
Clase: Padmasana, Imagen: File56.png, Formato: PNG
Clase: Padmasana, Imagen: File15.png, Formato: PNG
Clase: Padmasana, Imagen: File30.png, Formato: PNG
Clase: Padmasana, Imagen: File3.png, Formato: PNG
Clase: Padmasana, Imagen: File21.png, Formato: PNG
Clase: Padmasana, Imagen: File4.png, Formato: PNG
Clase: Padmasana, Imagen: File71.jpeg, Formato: JPEG
Clase: Padmasana, Imagen: File66.jpg, Formato: JPEG
Clase: Padmasana, Imagen: File24.png, Formato: PNG
Clase: Padmasana, Imagen: File67.jpg, Formato: JPEG
Clase: Padmasana, Imagen: File64.png, Formato: PNG
Clase: Padmasana, Imagen: File72.jpeg, Formato: JPEG
Clase: Padmasana, Imagen: File32.png, Formato: PNG
Clase: Padmasana, Imagen: File52.png, Formato: PNG
Clase: Padmasana, Imagen: File55.png, Formato: PNG
Clase: Padmasana, Imagen: File49.png, Formato: PNG
Clase: Padmasana, Imagen: File5.png, Formato: PNG
Clase: Padmasana, Imagen: File65.png, Formato: PNG
Clase: Padmasana, Imagen: File69.jpeg, Formato: JPEG
Clase: Padmasana, Imagen: File59.png, Formato: PNG
Clase: Padmasana, Imagen: File37.png, Formato: PNG
Clase: Padmasana, Imagen: File44.png, Formato: PNG
Clase: Padmasana, Imagen: File19.png, Formato: PNG
Clase: Padmasana, Imagen: File61.png, Formato: PNG
Clase: Padmasana, Imagen: File8.png, Formato: PNG
Clase: Padmasana, Imagen: File60.png, Formato: PNG
Clase: Padmasana, Imagen: File1.png, Formato: PNG
Clase: Padmasana, Imagen: File33.png, Formato: PNG
Clase: Padmasana, Imagen: File75.jpeg, Formato: JPEG
Clase: Padmasana, Imagen: File53.jpg, Formato: PNG
Clase: Padmasana, Imagen: File20.png, Formato: PNG

Clase: Padmasana, Imagen: File23.png, Formato: PNG
Clase: Padmasana, Imagen: File46.png, Formato: PNG
Clase: Padmasana, Imagen: File41.png, Formato: PNG
Clase: Padmasana, Imagen: File40.png, Formato: PNG
Clase: Padmasana, Imagen: File70.jpeg, Formato: JPEG
Clase: Padmasana, Imagen: File9.png, Formato: PNG
Clase: Padmasana, Imagen: File39.png, Formato: PNG
Clase: Padmasana, Imagen: File48.png, Formato: PNG
Clase: Padmasana, Imagen: File12.jpg, Formato: PNG
Clase: Padmasana, Imagen: File77.jpeg, Formato: JPEG
Clase: Padmasana, Imagen: File34.png, Formato: PNG
Clase: Padmasana, Imagen: File13.png, Formato: PNG
Clase: Padmasana, Imagen: File11.png, Formato: PNG
Clase: Padmasana, Imagen: File54.jpg, Formato: PNG
Clase: Padmasana, Imagen: File25.png, Formato: PNG
Clase: Padmasana, Imagen: File38.png, Formato: PNG
Clase: Padmasana, Imagen: File10.jpeg, Formato: PNG
Clase: Padmasana, Imagen: File31.png, Formato: PNG
Clase: Padmasana, Imagen: File35.png, Formato: PNG
Clase: Padmasana, Imagen: File2.png, Formato: PNG
Clase: Padmasana, Imagen: File57.png, Formato: PNG
Clase: Padmasana, Imagen: File68.png, Formato: PNG
Clase: Padmasana, Imagen: File51.png, Formato: PNG
Clase: Padmasana, Imagen: File36.png, Formato: PNG
Clase: Padmasana, Imagen: File7.png, Formato: PNG
Clase: Padmasana, Imagen: File47.png, Formato: PNG
Clase: Padmasana, Imagen: File14.png, Formato: PNG
Clase: Padmasana, Imagen: File6.png, Formato: PNG
Clase: Padmasana, Imagen: File43.jpg, Formato: PNG
Clase: Padmasana, Imagen: File18.png, Formato: PNG
Clase: Padmasana, Imagen: File27.png, Formato: PNG
Clase: Padmasana, Imagen: File22.png, Formato: PNG
Clase: Padmasana, Imagen: File42.png, Formato: PNG
Clase: Padmasana, Imagen: File63.png, Formato: PNG
Clase: Padmasana, Imagen: File76.jpeg, Formato: JPEG
Clase: Padmasana, Imagen: File62.png, Formato: PNG
Clase: Padmasana, Imagen: File50.png, Formato: PNG
Clase: Padmasana, Imagen: File58.png, Formato: PNG
Clase: Padmasana, Imagen: File74.jpeg, Formato: JPEG
Clase: Padmasana, Imagen: File26.png, Formato: PNG
Clase: Utthita Parsvakonasana, Imagen: File16.png, Formato: PNG
Clase: Utthita Parsvakonasana, Imagen: File17.png, Formato: PNG
Clase: Utthita Parsvakonasana, Imagen: File29.png, Formato: PNG
Clase: Utthita Parsvakonasana, Imagen: File28.png, Formato: PNG
Clase: Utthita Parsvakonasana, Imagen: File45.png, Formato: PNG
Clase: Utthita Parsvakonasana, Imagen: File56.png, Formato: PNG
Clase: Utthita Parsvakonasana, Imagen: File15.png, Formato: PNG
Clase: Utthita Parsvakonasana, Imagen: File30.png, Formato: PNG
Clase: Utthita Parsvakonasana, Imagen: File3.png, Formato: PNG
Clase: Utthita Parsvakonasana, Imagen: File21.png, Formato: PNG
Clase: Utthita Parsvakonasana, Imagen: File4.png, Formato: PNG
Clase: Utthita Parsvakonasana, Imagen: File24.png, Formato: PNG
Clase: Utthita Parsvakonasana, Imagen: File67.jpeg, Formato: JPEG
Clase: Utthita Parsvakonasana, Imagen: File53.png, Formato: PNG
Clase: Utthita Parsvakonasana, Imagen: File12.png, Formato: PNG
Clase: Utthita Parsvakonasana, Imagen: File32.png, Formato: PNG

[illegible]

[illegible]

Clase: Urdhva Mukha Svsnssana, Imagen: File18.png, Formato: PNG
Clase: Urdhva Mukha Svsnssana, Imagen: File27.png, Formato: PNG
Clase: Urdhva Mukha Svsnssana, Imagen: File22.png, Formato: PNG
Clase: Urdhva Mukha Svsnssana, Imagen: File42.png, Formato: PNG
Clase: Urdhva Mukha Svsnssana, Imagen: File63.png, Formato: PNG
Clase: Urdhva Mukha Svsnssana, Imagen: File62.png, Formato: PNG
Clase: Urdhva Mukha Svsnssana, Imagen: File43.png, Formato: PNG
Clase: Urdhva Mukha Svsnssana, Imagen: File50.png, Formato: PNG
Clase: Urdhva Mukha Svsnssana, Imagen: File58.png, Formato: PNG
Clase: Urdhva Mukha Svsnssana, Imagen: File26.png, Formato: PNG
Clase: Balasana, Imagen: File16.png, Formato: PNG
Clase: Balasana, Imagen: File45.jpg, Formato: PNG
Clase: Balasana, Imagen: File17.png, Formato: PNG
Clase: Balasana, Imagen: File29.png, Formato: PNG
Clase: Balasana, Imagen: File71.png, Formato: PNG
Clase: Balasana, Imagen: File28.png, Formato: PNG
Clase: Balasana, Imagen: File73.jpeg, Formato: JPEG
Clase: Balasana, Imagen: File56.png, Formato: PNG
Clase: Balasana, Imagen: File15.png, Formato: PNG
Clase: Balasana, Imagen: File79.jpeg, Formato: JPEG
Clase: Balasana, Imagen: File30.png, Formato: PNG
Clase: Balasana, Imagen: File3.png, Formato: PNG
Clase: Balasana, Imagen: File21.png, Formato: PNG
Clase: Balasana, Imagen: File4.png, Formato: PNG
Clase: Balasana, Imagen: File24.png, Formato: PNG
Clase: Balasana, Imagen: File53.png, Formato: PNG
Clase: Balasana, Imagen: File70.png, Formato: PNG
Clase: Balasana, Imagen: File64.png, Formato: PNG
Clase: Balasana, Imagen: File72.jpeg, Formato: JPEG
Clase: Balasana, Imagen: File12.png, Formato: PNG
Clase: Balasana, Imagen: File32.png, Formato: PNG
Clase: Balasana, Imagen: File52.jpg, Formato: JPEG
Clase: Balasana, Imagen: File54.png, Formato: PNG
Clase: Balasana, Imagen: File55.png, Formato: PNG
Clase: Balasana, Imagen: File49.png, Formato: PNG
Clase: Balasana, Imagen: File5.png, Formato: PNG
Clase: Balasana, Imagen: File65.png, Formato: PNG
Clase: Balasana, Imagen: File59.png, Formato: PNG
Clase: Balasana, Imagen: File37.png, Formato: PNG
Clase: Balasana, Imagen: File44.png, Formato: PNG
Clase: Balasana, Imagen: File19.png, Formato: PNG
Clase: Balasana, Imagen: File61.png, Formato: PNG
Clase: Balasana, Imagen: File8.png, Formato: PNG
Clase: Balasana, Imagen: File60.png, Formato: PNG
Clase: Balasana, Imagen: File1.png, Formato: PNG
Clase: Balasana, Imagen: File33.png, Formato: PNG
Clase: Balasana, Imagen: File75.jpeg, Formato: JPEG
Clase: Balasana, Imagen: File20.png, Formato: PNG
Clase: Balasana, Imagen: File23.png, Formato: PNG
Clase: Balasana, Imagen: File46.png, Formato: PNG
Clase: Balasana, Imagen: File41.png, Formato: PNG
Clase: Balasana, Imagen: File40.png, Formato: PNG
Clase: Balasana, Imagen: File9.png, Formato: PNG
Clase: Balasana, Imagen: File39.png, Formato: PNG
Clase: Balasana, Imagen: File48.png, Formato: PNG
Clase: Balasana, Imagen: File77.jpeg, Formato: JPEG

Clase: Balasana, Imagen: File34.png, Formato: PNG
Clase: Balasana, Imagen: File13.png, Formato: PNG
Clase: Balasana, Imagen: File10.png, Formato: PNG
Clase: Balasana, Imagen: File66.png, Formato: PNG
Clase: Balasana, Imagen: File67.png, Formato: PNG
Clase: Balasana, Imagen: File11.png, Formato: PNG
Clase: Balasana, Imagen: File25.png, Formato: PNG
Clase: Balasana, Imagen: File38.png, Formato: PNG
Clase: Balasana, Imagen: File31.png, Formato: PNG
Clase: Balasana, Imagen: File35.png, Formato: PNG
Clase: Balasana, Imagen: File2.png, Formato: PNG
Clase: Balasana, Imagen: File57.png, Formato: PNG
Clase: Balasana, Imagen: File68.png, Formato: PNG
Clase: Balasana, Imagen: File51.png, Formato: PNG
Clase: Balasana, Imagen: File69.png, Formato: PNG
Clase: Balasana, Imagen: File36.png, Formato: PNG
Clase: Balasana, Imagen: File7.png, Formato: PNG
Clase: Balasana, Imagen: File47.png, Formato: PNG
Clase: Balasana, Imagen: File14.png, Formato: PNG
Clase: Balasana, Imagen: File6.png, Formato: PNG
Clase: Balasana, Imagen: File18.png, Formato: PNG
Clase: Balasana, Imagen: File27.png, Formato: PNG
Clase: Balasana, Imagen: File22.png, Formato: PNG
Clase: Balasana, Imagen: File78.jpeg, Formato: JPEG
Clase: Balasana, Imagen: File42.png, Formato: PNG
Clase: Balasana, Imagen: File63.png, Formato: PNG
Clase: Balasana, Imagen: File62.jpg, Formato: PNG
Clase: Balasana, Imagen: File76.jpeg, Formato: JPEG
Clase: Balasana, Imagen: File43.png, Formato: PNG
Clase: Balasana, Imagen: File50.png, Formato: PNG
Clase: Balasana, Imagen: File58.png, Formato: PNG
Clase: Balasana, Imagen: File74.jpeg, Formato: JPEG
Clase: Balasana, Imagen: File26.png, Formato: PNG
Clase: Bitilasana, Imagen: File16.png, Formato: PNG
Clase: Bitilasana, Imagen: File17.png, Formato: PNG
Clase: Bitilasana, Imagen: File29.png, Formato: PNG
Clase: Bitilasana, Imagen: File71.png, Formato: PNG
Clase: Bitilasana, Imagen: File28.png, Formato: PNG
Clase: Bitilasana, Imagen: File45.png, Formato: PNG
Clase: Bitilasana, Imagen: File85.png, Formato: PNG
Clase: Bitilasana, Imagen: File56.png, Formato: PNG
Clase: Bitilasana, Imagen: File15.png, Formato: PNG
Clase: Bitilasana, Imagen: File30.png, Formato: PNG
Clase: Bitilasana, Imagen: File3.png, Formato: PNG
Clase: Bitilasana, Imagen: File75.png, Formato: PNG
Clase: Bitilasana, Imagen: File21.png, Formato: PNG
Clase: Bitilasana, Imagen: File4.png, Formato: PNG
Clase: Bitilasana, Imagen: File80.png, Formato: PNG
Clase: Bitilasana, Imagen: File74.png, Formato: PNG
Clase: Bitilasana, Imagen: File24.png, Formato: PNG
Clase: Bitilasana, Imagen: File53.png, Formato: PNG
Clase: Bitilasana, Imagen: File92.jpeg, Formato: JPEG
Clase: Bitilasana, Imagen: File70.png, Formato: PNG
Clase: Bitilasana, Imagen: File72.png, Formato: PNG
Clase: Bitilasana, Imagen: File86.png, Formato: PNG
Clase: Bitilasana, Imagen: File94.png, Formato: PNG

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

Clase: Utkatasana, Imagen: File5.png, Formato: PNG
Clase: Utkatasana, Imagen: File65.png, Formato: PNG
Clase: Utkatasana, Imagen: File59.png, Formato: PNG
Clase: Utkatasana, Imagen: File37.png, Formato: PNG
Clase: Utkatasana, Imagen: File44.png, Formato: PNG
Clase: Utkatasana, Imagen: File19.png, Formato: PNG
Clase: Utkatasana, Imagen: File61.png, Formato: PNG
Clase: Utkatasana, Imagen: File8.png, Formato: PNG
Clase: Utkatasana, Imagen: File60.png, Formato: PNG
Clase: Utkatasana, Imagen: File1.png, Formato: PNG
Clase: Utkatasana, Imagen: File33.png, Formato: PNG
Clase: Utkatasana, Imagen: File75.jpeg, Formato: JPEG
Clase: Utkatasana, Imagen: File20.png, Formato: PNG
Clase: Utkatasana, Imagen: File23.png, Formato: PNG
Clase: Utkatasana, Imagen: File46.png, Formato: PNG
Clase: Utkatasana, Imagen: File41.png, Formato: PNG
Clase: Utkatasana, Imagen: File40.png, Formato: PNG
Clase: Utkatasana, Imagen: File9.png, Formato: PNG
Clase: Utkatasana, Imagen: File39.png, Formato: PNG
Clase: Utkatasana, Imagen: File48.png, Formato: PNG
Clase: Utkatasana, Imagen: File77.jpeg, Formato: JPEG
Clase: Utkatasana, Imagen: File34.png, Formato: PNG
Clase: Utkatasana, Imagen: File13.png, Formato: PNG
Clase: Utkatasana, Imagen: File10.png, Formato: PNG
Clase: Utkatasana, Imagen: File66.png, Formato: PNG
Clase: Utkatasana, Imagen: File67.png, Formato: PNG
Clase: Utkatasana, Imagen: File11.png, Formato: PNG
Clase: Utkatasana, Imagen: File25.png, Formato: PNG
Clase: Utkatasana, Imagen: File38.png, Formato: PNG
Clase: Utkatasana, Imagen: File31.png, Formato: PNG
Clase: Utkatasana, Imagen: File35.png, Formato: PNG
Clase: Utkatasana, Imagen: File2.png, Formato: PNG
Clase: Utkatasana, Imagen: File57.png, Formato: PNG
Clase: Utkatasana, Imagen: File51.png, Formato: PNG
Clase: Utkatasana, Imagen: File81.png, Formato: PNG
Clase: Utkatasana, Imagen: File69.png, Formato: PNG
Clase: Utkatasana, Imagen: File36.png, Formato: PNG
Clase: Utkatasana, Imagen: File7.png, Formato: PNG
Clase: Utkatasana, Imagen: File47.png, Formato: PNG
Clase: Utkatasana, Imagen: File14.png, Formato: PNG
Clase: Utkatasana, Imagen: File6.png, Formato: PNG
Clase: Utkatasana, Imagen: File18.png, Formato: PNG
Clase: Utkatasana, Imagen: File27.png, Formato: PNG
Clase: Utkatasana, Imagen: File22.png, Formato: PNG
Clase: Utkatasana, Imagen: File42.jpg, Formato: PNG
Clase: Utkatasana, Imagen: File78.jpeg, Formato: JPEG
Clase: Utkatasana, Imagen: File63.png, Formato: PNG
Clase: Utkatasana, Imagen: File73.png, Formato: PNG
Clase: Utkatasana, Imagen: File76.jpeg, Formato: JPEG
Clase: Utkatasana, Imagen: File62.png, Formato: PNG
Clase: Utkatasana, Imagen: File43.png, Formato: PNG
Clase: Utkatasana, Imagen: File50.png, Formato: PNG
Clase: Utkatasana, Imagen: File58.png, Formato: PNG
Clase: Utkatasana, Imagen: File74.jpeg, Formato: JPEG
Clase: Utkatasana, Imagen: File26.png, Formato: PNG
Clase: Urdhva Dhanurasana, Imagen: File16.png, Formato: PNG

[illegible]

Clase: Urdhva Dhanurasana, Imagen: File68.png, Formato: PNG
Clase: Urdhva Dhanurasana, Imagen: File51.png, Formato: PNG
Clase: Urdhva Dhanurasana, Imagen: File36.png, Formato: PNG
Clase: Urdhva Dhanurasana, Imagen: File7.png, Formato: PNG
Clase: Urdhva Dhanurasana, Imagen: File47.png, Formato: PNG
Clase: Urdhva Dhanurasana, Imagen: File14.png, Formato: PNG
Clase: Urdhva Dhanurasana, Imagen: File6.png, Formato: PNG
Clase: Urdhva Dhanurasana, Imagen: File18.png, Formato: PNG
Clase: Urdhva Dhanurasana, Imagen: File27.png, Formato: PNG
Clase: Urdhva Dhanurasana, Imagen: File42.png, Formato: PNG
Clase: Urdhva Dhanurasana, Imagen: File63.png, Formato: PNG
Clase: Urdhva Dhanurasana, Imagen: File62.png, Formato: PNG
Clase: Urdhva Dhanurasana, Imagen: File43.png, Formato: PNG
Clase: Urdhva Dhanurasana, Imagen: File50.png, Formato: PNG
Clase: Urdhva Dhanurasana, Imagen: File58.png, Formato: PNG
Clase: Urdhva Dhanurasana, Imagen: File74.jpeg, Formato: JPEG
Clase: Urdhva Dhanurasana, Imagen: File26.png, Formato: PNG
Clase: Halasana, Imagen: File16.png, Formato: PNG
Clase: Halasana, Imagen: File17.png, Formato: PNG
Clase: Halasana, Imagen: File29.png, Formato: PNG
Clase: Halasana, Imagen: File28.png, Formato: PNG
Clase: Halasana, Imagen: File45.png, Formato: PNG
Clase: Halasana, Imagen: File56.png, Formato: PNG
Clase: Halasana, Imagen: File15.png, Formato: PNG
Clase: Halasana, Imagen: File30.png, Formato: PNG
Clase: Halasana, Imagen: File3.png, Formato: PNG
Clase: Halasana, Imagen: File21.png, Formato: PNG
Clase: Halasana, Imagen: File4.png, Formato: PNG
Clase: Halasana, Imagen: File71.jpeg, Formato: JPEG
Clase: Halasana, Imagen: File24.png, Formato: PNG
Clase: Halasana, Imagen: File67.jpeg, Formato: JPEG
Clase: Halasana, Imagen: File53.png, Formato: PNG
Clase: Halasana, Imagen: File64.png, Formato: PNG
Clase: Halasana, Imagen: File12.png, Formato: PNG
Clase: Halasana, Imagen: File32.png, Formato: PNG
Clase: Halasana, Imagen: File52.png, Formato: PNG
Clase: Halasana, Imagen: File54.png, Formato: PNG
Clase: Halasana, Imagen: File55.png, Formato: PNG
Clase: Halasana, Imagen: File49.png, Formato: PNG
Clase: Halasana, Imagen: File5.png, Formato: PNG
Clase: Halasana, Imagen: File65.png, Formato: PNG
Clase: Halasana, Imagen: File69.jpeg, Formato: JPEG
Clase: Halasana, Imagen: File59.png, Formato: PNG
Clase: Halasana, Imagen: File37.png, Formato: PNG
Clase: Halasana, Imagen: File44.png, Formato: PNG
Clase: Halasana, Imagen: File19.png, Formato: PNG
Clase: Halasana, Imagen: File61.png, Formato: PNG
Clase: Halasana, Imagen: File8.png, Formato: PNG
Clase: Halasana, Imagen: File60.png, Formato: PNG
Clase: Halasana, Imagen: File1.png, Formato: PNG
Clase: Halasana, Imagen: File33.png, Formato: PNG
Clase: Halasana, Imagen: File20.png, Formato: PNG
Clase: Halasana, Imagen: File23.png, Formato: PNG
Clase: Halasana, Imagen: File46.png, Formato: PNG
Clase: Halasana, Imagen: File41.png, Formato: PNG
Clase: Halasana, Imagen: File40.png, Formato: PNG

Clase: Halasana, Imagen: File70.jpeg, Formato: JPEG
Clase: Halasana, Imagen: File68.jpeg, Formato: JPEG
Clase: Halasana, Imagen: File9.png, Formato: PNG
Clase: Halasana, Imagen: File39.png, Formato: PNG
Clase: Halasana, Imagen: File48.png, Formato: PNG
Clase: Halasana, Imagen: File34.png, Formato: PNG
Clase: Halasana, Imagen: File13.png, Formato: PNG
Clase: Halasana, Imagen: File10.png, Formato: PNG
Clase: Halasana, Imagen: File66.png, Formato: PNG
Clase: Halasana, Imagen: File11.png, Formato: PNG
Clase: Halasana, Imagen: File25.png, Formato: PNG
Clase: Halasana, Imagen: File38.png, Formato: PNG
Clase: Halasana, Imagen: File31.png, Formato: PNG
Clase: Halasana, Imagen: File35.png, Formato: PNG
Clase: Halasana, Imagen: File2.png, Formato: PNG
Clase: Halasana, Imagen: File57.png, Formato: PNG
Clase: Halasana, Imagen: File51.png, Formato: PNG
Clase: Halasana, Imagen: File36.png, Formato: PNG
Clase: Halasana, Imagen: File7.png, Formato: PNG
Clase: Halasana, Imagen: File47.png, Formato: PNG
Clase: Halasana, Imagen: File14.png, Formato: PNG
Clase: Halasana, Imagen: File6.png, Formato: PNG
Clase: Halasana, Imagen: File18.png, Formato: PNG
Clase: Halasana, Imagen: File27.png, Formato: PNG
Clase: Halasana, Imagen: File22.png, Formato: PNG
Clase: Halasana, Imagen: File42.png, Formato: PNG
Clase: Halasana, Imagen: File63.png, Formato: PNG
Clase: Halasana, Imagen: File62.png, Formato: PNG
Clase: Halasana, Imagen: File43.png, Formato: PNG
Clase: Halasana, Imagen: File50.png, Formato: PNG
Clase: Halasana, Imagen: File58.png, Formato: PNG
Clase: Halasana, Imagen: File26.png, Formato: PNG
Clase: Ustrasana, Imagen: File16.png, Formato: PNG
Clase: Ustrasana, Imagen: File17.png, Formato: PNG
Clase: Ustrasana, Imagen: File29.png, Formato: PNG
Clase: Ustrasana, Imagen: File71.png, Formato: PNG
Clase: Ustrasana, Imagen: File28.png, Formato: PNG
Clase: Ustrasana, Imagen: File45.png, Formato: PNG
Clase: Ustrasana, Imagen: File85.png, Formato: PNG
Clase: Ustrasana, Imagen: File56.png, Formato: PNG
Clase: Ustrasana, Imagen: File15.png, Formato: PNG
Clase: Ustrasana, Imagen: File30.png, Formato: PNG
Clase: Ustrasana, Imagen: File3.png, Formato: PNG
Clase: Ustrasana, Imagen: File75.png, Formato: PNG
Clase: Ustrasana, Imagen: File21.png, Formato: PNG
Clase: Ustrasana, Imagen: File4.png, Formato: PNG
Clase: Ustrasana, Imagen: File80.png, Formato: PNG
Clase: Ustrasana, Imagen: File74.png, Formato: PNG
Clase: Ustrasana, Imagen: File24.png, Formato: PNG
Clase: Ustrasana, Imagen: File53.png, Formato: PNG
Clase: Ustrasana, Imagen: File92.jpeg, Formato: JPEG
Clase: Ustrasana, Imagen: File70.png, Formato: PNG
Clase: Ustrasana, Imagen: File72.png, Formato: PNG
Clase: Ustrasana, Imagen: File86.png, Formato: PNG
Clase: Ustrasana, Imagen: File64.png, Formato: PNG
Clase: Ustrasana, Imagen: File12.png, Formato: PNG

[illegible]

Clase: Ustrasana, Imagen: File14.png, Formato: PNG
Clase: Ustrasana, Imagen: File6.png, Formato: PNG
Clase: Ustrasana, Imagen: File95.jpeg, Formato: JPEG
Clase: Ustrasana, Imagen: File82.png, Formato: PNG
Clase: Ustrasana, Imagen: File18.png, Formato: PNG
Clase: Ustrasana, Imagen: File27.png, Formato: PNG
Clase: Ustrasana, Imagen: File22.png, Formato: PNG
Clase: Ustrasana, Imagen: File42.png, Formato: PNG
Clase: Ustrasana, Imagen: File63.png, Formato: PNG
Clase: Ustrasana, Imagen: File73.png, Formato: PNG
Clase: Ustrasana, Imagen: File62.png, Formato: PNG
Clase: Ustrasana, Imagen: File43.png, Formato: PNG
Clase: Ustrasana, Imagen: File93.jpeg, Formato: JPEG
Clase: Ustrasana, Imagen: File50.png, Formato: PNG
Clase: Ustrasana, Imagen: File58.png, Formato: PNG
Clase: Ustrasana, Imagen: File26.png, Formato: PNG
Clase: Alanasana, Imagen: File4.jpeg, Formato: JPEG
Clase: Alanasana, Imagen: File3.jpeg, Formato: JPEG
Clase: Alanasana, Imagen: File7.jpeg, Formato: JPEG
Clase: Alanasana, Imagen: File5.png, Formato: PNG
Clase: Alanasana, Imagen: File15.jpeg, Formato: JPEG
Clase: Alanasana, Imagen: File9.jpeg, Formato: JPEG
Clase: Alanasana, Imagen: File16.jpeg, Formato: JPEG
Clase: Alanasana, Imagen: File13.png, Formato: PNG
Clase: Alanasana, Imagen: File11.png, Formato: PNG
Clase: Alanasana, Imagen: File14.jpeg, Formato: JPEG
Clase: Alanasana, Imagen: File10.jpeg, Formato: JPEG
Clase: Alanasana, Imagen: File8.jpeg, Formato: JPEG
Clase: Alanasana, Imagen: File2.png, Formato: PNG
Clase: Alanasana, Imagen: File17.jpeg, Formato: JPEG
Clase: Alanasana, Imagen: File12.jpeg, Formato: JPEG
Clase: Alanasana, Imagen: File1.jpeg, Formato: JPEG
Clase: Alanasana, Imagen: File18.png, Formato: PNG
Clase: Alanasana, Imagen: File6.jpeg, Formato: JPEG
Clase: Parsvottanasana, Imagen: File16.png, Formato: PNG
Clase: Parsvottanasana, Imagen: File17.png, Formato: PNG
Clase: Parsvottanasana, Imagen: File29.png, Formato: PNG
Clase: Parsvottanasana, Imagen: File28.png, Formato: PNG
Clase: Parsvottanasana, Imagen: File42.jpeg, Formato: JPEG
Clase: Parsvottanasana, Imagen: File15.png, Formato: PNG
Clase: Parsvottanasana, Imagen: File30.png, Formato: PNG
Clase: Parsvottanasana, Imagen: File3.png, Formato: PNG
Clase: Parsvottanasana, Imagen: File21.png, Formato: PNG
Clase: Parsvottanasana, Imagen: File4.png, Formato: PNG
Clase: Parsvottanasana, Imagen: File24.png, Formato: PNG
Clase: Parsvottanasana, Imagen: File36.jpeg, Formato: JPEG
Clase: Parsvottanasana, Imagen: File12.png, Formato: PNG
Clase: Parsvottanasana, Imagen: File32.png, Formato: PNG
Clase: Parsvottanasana, Imagen: File39.jpeg, Formato: JPEG
Clase: Parsvottanasana, Imagen: File5.png, Formato: PNG
Clase: Parsvottanasana, Imagen: File41.jpeg, Formato: JPEG
Clase: Parsvottanasana, Imagen: File38.jpeg, Formato: JPEG
Clase: Parsvottanasana, Imagen: File37.jpeg, Formato: JPEG
Clase: Parsvottanasana, Imagen: File19.png, Formato: PNG
Clase: Parsvottanasana, Imagen: File8.png, Formato: PNG
Clase: Parsvottanasana, Imagen: File1.png, Formato: PNG

Clase: Parsvottanasana, Imagen: File33.png, Formato: PNG
Clase: Parsvottanasana, Imagen: File20.png, Formato: PNG
Clase: Parsvottanasana, Imagen: File23.png, Formato: PNG
Clase: Parsvottanasana, Imagen: File43.jpeg, Formato: JPEG
Clase: Parsvottanasana, Imagen: File9.png, Formato: PNG
Clase: Parsvottanasana, Imagen: File34.png, Formato: PNG
Clase: Parsvottanasana, Imagen: File13.png, Formato: PNG
Clase: Parsvottanasana, Imagen: File10.png, Formato: PNG
Clase: Parsvottanasana, Imagen: File11.png, Formato: PNG
Clase: Parsvottanasana, Imagen: File25.png, Formato: PNG
Clase: Parsvottanasana, Imagen: File40.jpeg, Formato: JPEG
Clase: Parsvottanasana, Imagen: File31.png, Formato: PNG
Clase: Parsvottanasana, Imagen: File35.png, Formato: PNG
Clase: Parsvottanasana, Imagen: File2.png, Formato: PNG
Clase: Parsvottanasana, Imagen: File7.png, Formato: PNG
Clase: Parsvottanasana, Imagen: File14.png, Formato: PNG
Clase: Parsvottanasana, Imagen: File6.png, Formato: PNG
Clase: Parsvottanasana, Imagen: File18.png, Formato: PNG
Clase: Parsvottanasana, Imagen: File27.png, Formato: PNG
Clase: Parsvottanasana, Imagen: File22.png, Formato: PNG
Clase: Parsvottanasana, Imagen: File26.png, Formato: PNG
Clase: Vrksasana, Imagen: File16.png, Formato: PNG
Clase: Vrksasana, Imagen: File17.png, Formato: PNG
Clase: Vrksasana, Imagen: File29.png, Formato: PNG
Clase: Vrksasana, Imagen: File28.png, Formato: PNG
Clase: Vrksasana, Imagen: File45.png, Formato: PNG
Clase: Vrksasana, Imagen: File56.png, Formato: PNG
Clase: Vrksasana, Imagen: File15.png, Formato: PNG
Clase: Vrksasana, Imagen: File30.png, Formato: PNG
Clase: Vrksasana, Imagen: File3.png, Formato: PNG
Clase: Vrksasana, Imagen: File21.png, Formato: PNG
Clase: Vrksasana, Imagen: File4.png, Formato: PNG
Clase: Vrksasana, Imagen: File24.png, Formato: PNG
Clase: Vrksasana, Imagen: File67.jpeg, Formato: JPEG
Clase: Vrksasana, Imagen: File53.png, Formato: PNG
Clase: Vrksasana, Imagen: File12.png, Formato: PNG
Clase: Vrksasana, Imagen: File32.png, Formato: PNG
Clase: Vrksasana, Imagen: File52.png, Formato: PNG
Clase: Vrksasana, Imagen: File54.png, Formato: PNG
Clase: Vrksasana, Imagen: File55.png, Formato: PNG
Clase: Vrksasana, Imagen: File64.jpeg, Formato: JPEG
Clase: Vrksasana, Imagen: File65.jpeg, Formato: JPEG
Clase: Vrksasana, Imagen: File49.png, Formato: PNG
Clase: Vrksasana, Imagen: File5.png, Formato: PNG
Clase: Vrksasana, Imagen: File59.png, Formato: PNG
Clase: Vrksasana, Imagen: File37.png, Formato: PNG
Clase: Vrksasana, Imagen: File44.png, Formato: PNG
Clase: Vrksasana, Imagen: File19.png, Formato: PNG
Clase: Vrksasana, Imagen: File61.png, Formato: PNG
Clase: Vrksasana, Imagen: File8.png, Formato: PNG
Clase: Vrksasana, Imagen: File60.png, Formato: PNG
Clase: Vrksasana, Imagen: File63.jpeg, Formato: JPEG
Clase: Vrksasana, Imagen: File1.png, Formato: PNG
Clase: Vrksasana, Imagen: File33.png, Formato: PNG
Clase: Vrksasana, Imagen: File66.jpeg, Formato: JPEG
Clase: Vrksasana, Imagen: File20.png, Formato: PNG

Clase: Vrksasana, Imagen: File23.png, Formato: PNG
Clase: Vrksasana, Imagen: File46.png, Formato: PNG
Clase: Vrksasana, Imagen: File41.png, Formato: PNG
Clase: Vrksasana, Imagen: File40.png, Formato: PNG
Clase: Vrksasana, Imagen: File9.png, Formato: PNG
Clase: Vrksasana, Imagen: File39.png, Formato: PNG
Clase: Vrksasana, Imagen: File48.png, Formato: PNG
Clase: Vrksasana, Imagen: File34.png, Formato: PNG
Clase: Vrksasana, Imagen: File13.png, Formato: PNG
Clase: Vrksasana, Imagen: File10.png, Formato: PNG
Clase: Vrksasana, Imagen: File11.png, Formato: PNG
Clase: Vrksasana, Imagen: File25.png, Formato: PNG
Clase: Vrksasana, Imagen: File38.png, Formato: PNG
Clase: Vrksasana, Imagen: File31.png, Formato: PNG
Clase: Vrksasana, Imagen: File35.png, Formato: PNG
Clase: Vrksasana, Imagen: File2.png, Formato: PNG
Clase: Vrksasana, Imagen: File57.png, Formato: PNG
Clase: Vrksasana, Imagen: File68.png, Formato: PNG
Clase: Vrksasana, Imagen: File51.png, Formato: PNG
Clase: Vrksasana, Imagen: File36.png, Formato: PNG
Clase: Vrksasana, Imagen: File7.png, Formato: PNG
Clase: Vrksasana, Imagen: File47.png, Formato: PNG
Clase: Vrksasana, Imagen: File14.png, Formato: PNG
Clase: Vrksasana, Imagen: File6.png, Formato: PNG
Clase: Vrksasana, Imagen: File18.png, Formato: PNG
Clase: Vrksasana, Imagen: File27.png, Formato: PNG
Clase: Vrksasana, Imagen: File22.png, Formato: PNG
Clase: Vrksasana, Imagen: File42.png, Formato: PNG
Clase: Vrksasana, Imagen: File62.png, Formato: PNG
Clase: Vrksasana, Imagen: File43.png, Formato: PNG
Clase: Vrksasana, Imagen: File50.png, Formato: PNG
Clase: Vrksasana, Imagen: File58.png, Formato: PNG
Clase: Vrksasana, Imagen: File26.png, Formato: PNG
Clase: Garudasana, Imagen: File16.png, Formato: PNG
Clase: Garudasana, Imagen: File17.png, Formato: PNG
Clase: Garudasana, Imagen: File29.png, Formato: PNG
Clase: Garudasana, Imagen: File71.png, Formato: PNG
Clase: Garudasana, Imagen: File28.png, Formato: PNG
Clase: Garudasana, Imagen: File45.png, Formato: PNG
Clase: Garudasana, Imagen: File56.png, Formato: PNG
Clase: Garudasana, Imagen: File80.jpeg, Formato: JPEG
Clase: Garudasana, Imagen: File15.png, Formato: PNG
Clase: Garudasana, Imagen: File79.jpeg, Formato: JPEG
Clase: Garudasana, Imagen: File30.png, Formato: PNG
Clase: Garudasana, Imagen: File3.png, Formato: PNG
Clase: Garudasana, Imagen: File75.png, Formato: PNG
Clase: Garudasana, Imagen: File21.png, Formato: PNG
Clase: Garudasana, Imagen: File4.png, Formato: PNG
Clase: Garudasana, Imagen: File74.png, Formato: PNG
Clase: Garudasana, Imagen: File24.png, Formato: PNG
Clase: Garudasana, Imagen: File53.png, Formato: PNG
Clase: Garudasana, Imagen: File81.jpeg, Formato: JPEG
Clase: Garudasana, Imagen: File70.png, Formato: PNG
Clase: Garudasana, Imagen: File72.png, Formato: PNG
Clase: Garudasana, Imagen: File64.png, Formato: PNG
Clase: Garudasana, Imagen: File12.png, Formato: PNG

[illegible]

[illegible]

Clase: Paschimottanasana, Imagen: File36.png, Formato: PNG
Clase: Paschimottanasana, Imagen: File7.png, Formato: PNG
Clase: Paschimottanasana, Imagen: File47.png, Formato: PNG
Clase: Paschimottanasana, Imagen: File14.png, Formato: PNG
Clase: Paschimottanasana, Imagen: File6.png, Formato: PNG
Clase: Paschimottanasana, Imagen: File18.png, Formato: PNG
Clase: Paschimottanasana, Imagen: File27.png, Formato: PNG
Clase: Paschimottanasana, Imagen: File22.png, Formato: PNG
Clase: Paschimottanasana, Imagen: File42.png, Formato: PNG
Clase: Paschimottanasana, Imagen: File43.png, Formato: PNG
Clase: Paschimottanasana, Imagen: File50.png, Formato: PNG
Clase: Paschimottanasana, Imagen: File26.png, Formato: PNG

Se observa que tenemos 2 diferentes tipos de imágenes: png y jpeg.
Es importante que antes de entrenar los modelos todas las imágenes
tengan el mismo modo de color.

Analizar etiquetas o metadata

```
In [ ]: def analizar_metadata(metadata_file):  
        metadata = pd.read_json(metadata_file)  
        return metadata  
  
        # Ruta del archivo JSON de metadata  
metadata = analizar_metadata(f'{folder_path}/Poses.json')  
print(metadata)
```


Poses

```

0  {'id': 1, 'sanskrit_name': 'Navasana', 'englis...
1  {'id': 2, 'sanskrit_name': 'Ardha Navasana', '...
2  {'id': 3, 'sanskrit_name': 'Dhanurasana', 'eng...
3  {'id': 4, 'sanskrit_name': 'Setu Bandha Sarvan...
4  {'id': 5, 'sanskrit_name': 'Baddha Konasana', ...
5  {'id': 6, 'sanskrit_name': 'Ustrasana', 'engli...
6  {'id': 7, 'sanskrit_name': 'Marjaryasana', 'en...
7  {'id': 8, 'sanskrit_name': 'Bitilasana', 'engl...
8  {'id': 9, 'sanskrit_name': 'Utkatasana', 'engl...
9  {'id': 10, 'sanskrit_name': 'Balasana', 'engli...
10 {'id': 11, 'sanskrit_name': 'Sivasana', 'engli...
11 {'id': 12, 'sanskrit_name': 'Alanasana', 'engl...
12 {'id': 13, 'sanskrit_name': 'Bakasana', 'engli...
13 {'id': 14, 'sanskrit_name': 'Ardha Pincha Mayu...
14 {'id': 15, 'sanskrit_name': 'Adho Mukha Svanas...
15 {'id': 16, 'sanskrit_name': 'Garudasana', 'eng...
16 {'id': 17, 'sanskrit_name': 'Utthita Hasta Pad...
17 {'id': 18, 'sanskrit_name': 'Utthita Parsvakon...
18 {'id': 19, 'sanskrit_name': 'Pincha Mayurasana...
19 {'id': 20, 'sanskrit_name': 'Uttanasana', 'eng...
20 {'id': 21, 'sanskrit_name': 'Ardha Chandrasana...
21 {'id': 22, 'sanskrit_name': 'Adho Mukha Vrksas...
22 {'id': 23, 'sanskrit_name': 'Anjaneyasana', 'e...
23 {'id': 24, 'sanskrit_name': 'Supta Kapotasana'...
24 {'id': 25, 'sanskrit_name': 'Eka Pada Rajakapo...
25 {'id': 26, 'sanskrit_name': 'Phalakasana', 'en...
26 {'id': 27, 'sanskrit_name': 'Halasana', 'engli...
27 {'id': 28, 'sanskrit_name': 'Parsvottanasana',...
28 {'id': 29, 'sanskrit_name': 'Parsva Virabhadra...
29 {'id': 30, 'sanskrit_name': 'Paschimottanasana...
30 {'id': 31, 'sanskrit_name': 'Padmasana', 'engl...
31 {'id': 32, 'sanskrit_name': 'Ardha Matsyendras...
32 {'id': 33, 'sanskrit_name': 'Salamba Sarvangas...
33 {'id': 34, 'sanskrit_name': 'Vasisthasana', 'e...
34 {'id': 35, 'sanskrit_name': 'Salamba Bhujangas...
35 {'id': 36, 'sanskrit_name': 'Hanumanasana', 'e...
36 {'id': 37, 'sanskrit_name': 'Malasana', 'engli...
37 {'id': 38, 'sanskrit_name': 'Uttanasana', 'eng...
38 {'id': 39, 'sanskrit_name': 'Ashta Chandrasana...
39 {'id': 40, 'sanskrit_name': 'Upavistha Konasan...
40 {'id': 41, 'sanskrit_name': 'Vrksasana', 'engl...
41 {'id': 42, 'sanskrit_name': 'Trikonasana', 'en...
42 {'id': 43, 'sanskrit_name': 'Urdhva Mukha Svsn...
43 {'id': 44, 'sanskrit_name': 'Virabhadrasana On...
44 {'id': 45, 'sanskrit_name': 'Virabhadrasana Tw...
45 {'id': 46, 'sanskrit_name': 'Virabhadrasana Th...
46 {'id': 47, 'sanskrit_name': 'Urdhva Dhanurasan...
47 {'id': 48, 'sanskrit_name': 'Camatkarasana', '...'

```

En los metadata, vemos para cada una de las poses:

```

{
  id:
  sanskrit_name:
  english_name:

```

```
    img_url:  
}
```

Acondicionamiento del Dataset (Preprocesamiento)

En este apartado se realizan las siguientes acciones para preprocesar el dataset:

1. Etiquetado: Asignar etiquetas a las imágenes.
2. Conversión de Formato: Convertir las imágenes a un formato estándar (RGB, sin transparencia).
3. Recorte o Padding: Si es necesario, agregar relleno o recortar las imágenes
4. Redimensionado: Asegurarse de que todas las imágenes tengan las mismas dimensiones. Los factores a considerar para determinar el tamaño óptimo son:
 - Tamaño de la red neuronal que se va a usar:
 - Modelos preentrenados: depende del tamaño específico de imagen con el que se ha preentrenado el modelo usado. Por lo tanto, es recomendable redimensionar las imágenes a ese tamaño si se utiliza transfer learning.
 - Modelos personalizados: En un modelo desde cero, el tamaño de la imagen depende del equilibrio entre precisión y tiempo de entrenamiento. Imágenes más grandes pueden mejorar el desempeño del modelo, pero a costa de un mayor tiempo de entrenamiento y más memoria.
 - Memoria y recursos computacionales:
 - Las imágenes de mayor resolución requieren más memoria (RAM y VRAM si se usa GPUs), lo cual puede ralentizar el proceso de entrenamiento.
 - Si se está limitado por los recursos computacionales, redimensionar las imágenes a un tamaño más pequeño puede ayudar a acelerar el entrenamiento. Sin embargo, imágenes demasiado pequeñas pueden perder detalles importantes que podrían afectar el rendimiento del modelo.

- Tarea específica:
 - Clasificación de imágenes: Si el dataset tiene detalles finos que son importantes para la clasificación (por ejemplo, reconocer pequeñas diferencias entre clases similares), sería mejor mantener una resolución más alta.
 - Reconocimiento de objetos: Si la tarea implica la localización o detección de objetos, también puede ser útil mantener una resolución más alta para que los objetos pequeños sean detectables.
 - Reducción de dimensionalidad o tareas rápidas: Si la precisión no es tan crítica y solo se desea entrenar un modelo rápido, se puede optar por tamaños más pequeños.
- Consideración sobre las proporciones:
 - Las imágenes deben tener una proporción consistente para que la red neuronal pueda procesarlas correctamente. Si las imágenes no son cuadradas, generalmente se redimensionan a un tamaño cuadrado (por ejemplo, 224x224, 299x299) o se recortan centralmente.

5. Normalización: Escalar los valores de los píxeles.

6. Aumentación de Datos: Aplicar transformaciones aleatorias para aumentar la variedad de datos y crear datos para balancear las clases

7. Guardar el Dataset

```
In [ ]: # variables para seleccion de tratamiento de dataset segun necesidades y apl
```

```
NORMALIZAR = '[0,1]' # [-1,1]
size = (224, 224)
color_fondo = (255, 255, 255) # Fondo blanco para el padding
SAVE_IMAGES = True # Cambia a True si quieres guardar
save_dir = os.path.join(BASE_FOLDER, "final")
```

```
In [ ]: def convertir_a_rgb_con_fondo(imagen_path, color_fondo=(255, 255, 255)):
    """
    Convierte la imagen a RGB con fondo sólido solo si tiene transparencia.
    Si la imagen no tiene transparencia, simplemente la convierte a RGB.
    :param imagen_path: Ruta de la imagen original.
    :param color_fondo: Color de fondo a usar si la imagen tiene transparenc
    :return: Imagen en formato RGB con fondo sólido si era necesario.
    """
    # Abrir la imagen
```

```

img = Image.open(imagen_path)

# Verificar si la imagen tiene un canal alfa (transparencia)
if img.mode == 'RGBA':
    # Si tiene un canal alfa (transparente), crear una nueva imagen con
    fondo = Image.new('RGBA', img.size, color_fondo + (255,)) # Color d
    fondo.paste(img, (0, 0), img) # Pegar la imagen original en la nuev
    img = fondo.convert('RGB') # Convertir a RGB, eliminando el canal a
elif img.mode == 'P': # Si la imagen está en modo "P" (paleta de colore
    img = img.convert('RGB')
else:
    # Si la imagen ya está en RGB o en otro formato sin transparencia, s
    img = img.convert('RGB')

return img

def hacer_cuadrada_con_padding(img, color_fondo=(255, 255, 255)):
    """Convierte una imagen rectangular en una imagen cuadrada con padding"""
    ancho, alto = img.size
    max_lado = max(ancho, alto)

    # Crear imagen cuadrada con fondo sólido
    nueva_img = Image.new("RGB", (max_lado, max_lado), color_fondo)
    offset = ((max_lado - ancho) // 2, (max_lado - alto) // 2)
    nueva_img.paste(img, offset)
    return nueva_img

# 1. Etiquetado con los metadatos de JSON
with open("DL/yoga_dataset/Poses.json") as f:
    data = json.load(f)

# Crear el diccionario de etiquetas
label_map = {pose["sanskrit_name"]: pose["id"] for pose in data["Poses"]}

# Definir la ruta del directorio con las imágenes
dataset_dir = os.path.join(BASE_FOLDER, "yoga_dataset")

# Lista para almacenar las imágenes y las etiquetas
images = []
labels = []

# Recorrer todas las carpetas de las clases
for class_name in os.listdir(dataset_dir):
    class_dir = os.path.join(dataset_dir, class_name)

    if os.path.isdir(class_dir): # Verificar si es un directorio
        label = label_map.get(class_name) # Obtener la etiqueta para esta c

        if label is None:
            print(f"Etiqueta no encontrada para {class_name}, saltando esta
                continue

        # Recorrer las imágenes de la clase
        for filename in os.listdir(class_dir):

```

```

image_path = os.path.join(class_dir, filename)

if image_path.endswith((''.jpg', '.jpeg', '.png')): # Verificar
    # 2. Convertir la imagen a RGB con fondo sólido
    imagen = convertir_a_rgb_con_fondo(image_path)

    # 3. Convertir la imagen a cuadrada con padding
    imagen_cuadrada = hacer_cuadrada_con_padding(imagen, color_f

    # 4. Redimensionar la imagen
    img_resized = imagen_cuadrada.resize(size)

    # Convertir la imagen a un arreglo NumPy
    imagen_array = np.array(img_resized)

    # 5. Normalizar la imagen (si es necesario)
    if NORMALIZAR == '[0,1]':
        imagen_array = imagen_array / 255.0 # Normalizar a rango
    elif NORMALIZAR == '[-1,1]':
        imagen_array = (imagen_array / 127.5) - 1.0

    # 6. Almacenar la imagen y la etiqueta
    images.append(imagen_array)
    labels.append(label)

# Convertir las listas en arrays numpy para usarlas en el modelo
images = np.array(images)
labels = np.array(labels)

```

```

/usr/local/lib/python3.11/dist-packages/PIL/Image.py:1043: UserWarning: Palette
te images with Transparency expressed in bytes should be converted to RGBA im
ages
  warnings.warn(

```

```

In [ ]: # --- AGRUPAR IMÁGENES POR CLASE ---
        imagenes_por_clase = defaultdict(list)
        for img, lbl in zip(images, labels):
            imagenes_por_clase[lbl].append(img)

        # --- OBTENER CLASE MÁS NUMEROSA ---
        max_count = max(len(imgs) for imgs in imagenes_por_clase.values())

```

```

In [ ]: # --- CONFIGURAR DATA AUGMENTATION ---
        datagen = ImageDataGenerator(
            rotation_range=30,
            width_shift_range=0.2,
            height_shift_range=0.2,
            shear_range=0.2,
            zoom_range=0.2,
            horizontal_flip=True,
            fill_mode='nearest'
        )

        # --- CREAR CARPETA DE SALIDA POR CLASE ---
        if SAVE_IMAGES:
            os.makedirs(save_dir, exist_ok=True)

```

```

for label, imagenes in imagenes_por_clase.items():
    imagenes = np.array(imagenes)
    original_count = len(imagenes)
    deficit = max_count - original_count

    # Crear subcarpeta para la clase
    if SAVE_IMAGES:
        class_dir = os.path.join(save_dir, f"class_{label}")
        os.makedirs(class_dir, exist_ok=True)

    # Guardar imágenes originales
    for i, img in enumerate(imagenes):
        if SAVE_IMAGES:
            img_pil = Image.fromarray((img * 255).astype(np.uint8))
            img_pil.save(os.path.join(class_dir, f"original_{i}.jpeg"))

    # Generar y guardar imágenes aumentadas
    if deficit > 0:
        gen = datagen.flow(imagenes, batch_size=1)
        for i in range(deficit):
            img_aug = next(gen)[0]
            if SAVE_IMAGES:
                img_pil = Image.fromarray((img_aug * 255).astype(np.uint8))
                img_pil.save(os.path.join(class_dir, f"aug_{i}.jpeg"))

print("Proceso de balanceo y guardado completo.")

```

Proceso de balanceo y guardado completo.

```

In [ ]: conteo = {}

for class_folder in os.listdir(save_dir):
    class_path = os.path.join(save_dir, class_folder)
    if os.path.isdir(class_path):
        num_imgs = len([f for f in os.listdir(class_path) if f.lower().endsw
        conteo[class_folder] = num_imgs

# Mostrar resultados
print("Conteo de imágenes por clase:")
for clase in sorted(conteo):
    print(f"{clase}: {conteo[clase]} imágenes")

min_count = min(conteo.values())
max_count = max(conteo.values())
if min_count == max_count:
    print(" El dataset está BALANCEADO.")
else:
    print(f" El dataset NO está balanceado. Min: {min_count}, Max: {max_cou

```

Conteo de imágenes por clase:

```
class_1: 96 imágenes
class_10: 96 imágenes
class_11: 96 imágenes
class_12: 96 imágenes
class_13: 96 imágenes
class_14: 96 imágenes
class_15: 96 imágenes
class_16: 96 imágenes
class_17: 96 imágenes
class_18: 96 imágenes
class_19: 96 imágenes
class_2: 96 imágenes
class_21: 96 imágenes
class_22: 96 imágenes
class_23: 96 imágenes
class_24: 96 imágenes
class_25: 96 imágenes
class_26: 96 imágenes
class_27: 96 imágenes
class_28: 96 imágenes
class_29: 96 imágenes
class_3: 96 imágenes
class_30: 96 imágenes
class_31: 96 imágenes
class_32: 96 imágenes
class_33: 96 imágenes
class_34: 96 imágenes
class_35: 96 imágenes
class_36: 96 imágenes
class_37: 96 imágenes
class_38: 96 imágenes
class_39: 96 imágenes
class_4: 96 imágenes
class_40: 96 imágenes
class_41: 96 imágenes
class_42: 96 imágenes
class_43: 96 imágenes
class_44: 96 imágenes
class_45: 96 imágenes
class_46: 96 imágenes
class_47: 96 imágenes
class_48: 96 imágenes
class_5: 96 imágenes
class_6: 96 imágenes
class_7: 96 imágenes
class_8: 96 imágenes
class_9: 96 imágenes
```

□ El dataset está BALANCEADO.

Al llegar a este punto, tenemos un dataset balanceado y preprocesado al que se le ha aplicado los puntos descritos anteriormente.

Es posible que en función de cada estrategia empleada, sea necesario aplicar unas características diferentes al preprocesamiento del

dataset.

Estrategia 1: Entrenamiento desde Cero

En esta sección se entrena una red neuronal convolucional (CNN) propia, diseñada desde cero, para clasificar las imágenes del conjunto de posturas de yoga. La arquitectura implementa varias técnicas de regularización como Dropout, Batch Normalization y penalización L1/L2 en los pesos de las capas convolucionales, con el fin de mejorar la generalización del modelo.

El modelo se compila, entrena y se monitoriza su evolución mediante métricas de precisión y pérdida sobre los conjuntos de entrenamiento y validación. A continuación, se presenta una tabla con los pasos y código correspondiente a modo de resumen.

Paso	Código correspondiente
Diseñar CNN propia	<code>Sequential([... Conv2D, MaxPooling2D, Dense ...])</code>
Aplicar regularización	<code>Dropout()</code> , <code>BatchNormalization()</code>
Compilar el modelo	<code>model.compile(optimizer='adam', loss=..., metrics=...)</code>
Entrenar y monitorizar	<code>model.fit(...)</code> + <code>callbacks</code> + <code>plt.plot(history.history)</code>

Primero instalo las librerías necesarias.

```
In [ ]: !pip install tensorflow
```


Requirement already satisfied: tensorflow in /usr/local/lib/python3.11/dist-packages (2.18.0)

Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.4.0)

Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.6.3)

Requirement already satisfied: flatbuffers>=24.3.25 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (25.2.10)

Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (0.6.0)

Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (0.2.0)

Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (18.1.1)

Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (3.4.0)

Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from tensorflow) (24.2)

Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<6.0.0dev,>=3.20.3 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (5.29.4)

Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (2.32.3)

Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-packages (from tensorflow) (75.2.0)

Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.17.0)

Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (3.1.0)

Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (4.13.2)

Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.17.2)

Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.71.0)

Requirement already satisfied: tensorboard<2.19,>=2.18 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (2.18.0)

Requirement already satisfied: keras>=3.5.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (3.8.0)

Requirement already satisfied: numpy<2.1.0,>=1.26.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (2.0.2)

Requirement already satisfied: h5py>=3.11.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (3.13.0)

Requirement already satisfied: ml-dtypes<0.5.0,>=0.4.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (0.4.1)

Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (0.37.1)

Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.11/dist-packages (from astunparse>=1.6.0->tensorflow) (0.45.1)

Requirement already satisfied: rich in /usr/local/lib/python3.11/dist-packages (from keras>=3.5.0->tensorflow) (13.9.4)

Requirement already satisfied: namex in /usr/local/lib/python3.11/dist-packages (from keras>=3.5.0->tensorflow) (0.0.9)

Requirement already satisfied: optree in /usr/local/lib/python3.11/dist-packages (from keras>=3.5.0->tensorflow) (0.15.0)

Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/pyt

hon3.11/dist-packages (from requests<3,>=2.21.0->tensorflow) (3.4.2)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensorflow) (3.10)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensorflow) (2.4.0)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensorflow) (2025.4.26)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.11/dist-packages (from tensorboard<2.19,>=2.18->tensorflow) (3.8)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.11/dist-packages (from tensorboard<2.19,>=2.18->tensorflow) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from tensorboard<2.19,>=2.18->tensorflow) (3.1.3)
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.11/dist-packages (from werkzeug>=1.0.1->tensorboard<2.19,>=2.18->tensorflow) (3.0.2)
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.11/dist-packages (from rich->keras>=3.5.0->tensorflow) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from rich->keras>=3.5.0->tensorflow) (2.19.1)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-packages (from markdown-it-py>=2.2.0->rich->keras>=3.5.0->tensorflow) (0.1.2)

```
In [ ]: from tensorflow.keras.models import Model, Sequential
        from tensorflow.keras.layers import GlobalAveragePooling2D, Dense, Dropout,
        from tensorflow.keras.optimizers import Adam
        from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
        import cv2
```

Cargo y balanceo las imágenes para esta estrategia.

```
In [ ]: import os
        import numpy as np

        # Ruta correcta
        save_dir = '/content/DL/final'

        balanced_images = []
        balanced_labels = []
        label_map = {}

        for idx, class_name in enumerate(sorted(os.listdir(save_dir))):
            class_path = os.path.join(save_dir, class_name)
            if os.path.isdir(class_path):
                label_map[idx] = class_name
                for img_file in os.listdir(class_path):
                    if img_file.lower().endswith(('.jpg', '.jpeg', '.png')):
                        img_path = os.path.join(class_path, img_file)
                        img = cv2.imread(img_path)
                        img = cv2.resize(img, (224, 224))
                        balanced_images.append(img)
                        balanced_labels.append(idx)
```

```
balanced_images = np.array(balanced_images) / 255.0 # Normalizar
balanced_labels = np.array(balanced_labels)
```

Compilación y entrenamiento del modelo

En este bloque se realiza la división del conjunto de datos en entrenamiento y validación, utilizando la función `train_test_split` de `scikit-learn`. Se asigna el 80% de las imágenes al conjunto de entrenamiento y el 20% al de validación, manteniendo la proporción original de clases mediante el parámetro `stratify`. Esto permite evaluar el rendimiento del modelo con datos no vistos durante el entrenamiento.

```
In [ ]: from sklearn.model_selection import train_test_split

X_train, X_val, y_train, y_val = train_test_split(
    balanced_images, balanced_labels,
    test_size=0.2,
    stratify=balanced_labels,
    random_state=42
)
```

```
In [ ]: train_datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)
```

```
In [ ]: BASE_FOLDER = "/content/DL"
save_dir = os.path.join(BASE_FOLDER, "final")
```

```
In [ ]: train_datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2
)

train_generator = train_datagen.flow_from_directory(
    directory=save_dir,
    target_size=(224, 224),
    batch_size=32,
    class_mode='sparse',
    subset='training'
)

val_generator = train_datagen.flow_from_directory(
```

```

    directory=save_dir,
    target_size=(224, 224),
    batch_size=32,
    class_mode='sparse',
    subset='validation'
)

```

Found 3619 images belonging to 47 classes.

Found 893 images belonging to 47 classes.

En esta estrategia se hace una red neuronal convolucional basada en una arquitectura secuencial con el objetivo de equilibrar rendimiento y simplicidad, considerando los recursos limitados de Google Colab. Esta se ha definido a mano usando Sequential en Keras.

Esta red CNN se crea desde 0 sin usar modelos preentrenados con varias capas Conv2D, MaxPooling2D, Flatten y Dense. Como técnica de regularización se ha aplicado Dropout y BatchNormalization.

Se utiliza Conv2D + Relu como activación para extraer características visuales de las imágenes (bordes, formas, texturas). El MaxPooling2D se escoge para reducir la dimensión de las imágenes y evitar el sobreajuste.

```

In [ ]: model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 3)),
    BatchNormalization(),
    MaxPooling2D(pool_size=(2, 2)),

    Conv2D(64, (3, 3), activation='relu'),
    BatchNormalization(),
    MaxPooling2D(pool_size=(2, 2)),

    Conv2D(128, (3, 3), activation='relu'),
    BatchNormalization(),
    MaxPooling2D(pool_size=(2, 2)),

    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.3),
    Dense(47, activation='softmax')
])

```

```

/usr/local/lib/python3.11/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.

```

```

    super().__init__(activity_regularizer=activity_regularizer, **kwargs)

```

El modelo se compila utilizando el **optimizador Adam**, con una tasa de **aprendizaje estándar**, y la **función de pérdida**

sparse_categorical_crossentropy, adecuada para **clasificación multiclase con etiquetas enteras**.

Se emplean callbacks como **EarlyStopping** para detener el entrenamiento si la validación no mejora, y **ReduceLROnPlateau** para ajustar automáticamente la tasa de aprendizaje. Esto permite un entrenamiento más eficiente y adaptativo, monitorizando la pérdida y precisión en tiempo real.

```
In [ ]: model.compile(
    optimizer=Adam(learning_rate=0.0005),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

model.summary()
```

Model: "sequential_3"

Layer (type)	Output Shape	Param
conv2d_216 (Conv2D)	(None, 222, 222, 32)	
batch_normalization_221 (BatchNormalization)	(None, 222, 222, 32)	
max_pooling2d_13 (MaxPooling2D)	(None, 111, 111, 32)	
conv2d_217 (Conv2D)	(None, 109, 109, 64)	18
batch_normalization_222 (BatchNormalization)	(None, 109, 109, 64)	
max_pooling2d_14 (MaxPooling2D)	(None, 54, 54, 64)	
conv2d_218 (Conv2D)	(None, 52, 52, 128)	73
batch_normalization_223 (BatchNormalization)	(None, 52, 52, 128)	
max_pooling2d_15 (MaxPooling2D)	(None, 26, 26, 128)	
flatten_3 (Flatten)	(None, 86528)	
dense_16 (Dense)	(None, 128)	11,075
dropout_13 (Dropout)	(None, 128)	
dense_17 (Dense)	(None, 47)	6

Total params: 11,175,919 (42.63 MB)

Trainable params: 11,175,471 (42.63 MB)

Non-trainable params: 448 (1.75 KB)

```
In [ ]: early_stop = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
        reduce_lr = ReduceLRonPlateau(monitor='val_loss', patience=3, factor=0.5)

        history = model.fit(
            train_generator,
            epochs=20,
            validation_data=val_generator,
            callbacks=[early_stop, reduce_lr]
        )
```

Epoch 1/20

```
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as they will be ignored.
  self.warn_if_super_not_called()
```

```
114/114 ████████████████████████████████ 14s 77ms/step - accuracy: 0.0514 - loss: 5.2230 - val_accuracy: 0.0213 - val_loss: 52.0827 - learning_rate: 5.0000e-04
```

Epoch 2/20

```
114/114 ████████████████████████████████ 5s 44ms/step - accuracy: 0.0535 - loss: 3.7904 - val_accuracy: 0.0213 - val_loss: 49.7143 - learning_rate: 5.0000e-04
```

Epoch 3/20

```
114/114 ████████████████████████████████ 5s 46ms/step - accuracy: 0.0695 - loss: 3.7179 - val_accuracy: 0.0269 - val_loss: 16.2774 - learning_rate: 5.0000e-04
```

Epoch 4/20

```
114/114 ████████████████████████████████ 5s 43ms/step - accuracy: 0.0876 - loss: 3.6418 - val_accuracy: 0.0235 - val_loss: 11.9406 - learning_rate: 5.0000e-04
```

Epoch 5/20

```
114/114 ████████████████████████████████ 5s 42ms/step - accuracy: 0.1337 - loss: 3.4399 - val_accuracy: 0.0515 - val_loss: 4.1126 - learning_rate: 5.0000e-04
```

Epoch 6/20

```
114/114 ████████████████████████████████ 5s 41ms/step - accuracy: 0.1443 - loss: 3.3483 - val_accuracy: 0.0459 - val_loss: 3.9862 - learning_rate: 5.0000e-04
```

Epoch 7/20

```
114/114 ████████████████████████████████ 5s 41ms/step - accuracy: 0.1720 - loss: 3.1820 - val_accuracy: 0.0336 - val_loss: 4.0189 - learning_rate: 5.0000e-04
```

Epoch 8/20

```
114/114 ████████████████████████████████ 5s 43ms/step - accuracy: 0.2097 - loss: 3.0490 - val_accuracy: 0.0594 - val_loss: 4.4956 - learning_rate: 5.0000e-04
```

Epoch 9/20

```
114/114 ████████████████████████████████ 5s 39ms/step - accuracy: 0.2252 - loss: 2.9565 - val_accuracy: 0.0784 - val_loss: 5.0656 - learning_rate: 5.0000e-04
```

Epoch 10/20

```
114/114 ████████████████████████████████ 5s 42ms/step - accuracy: 0.2416 - loss: 2.8722 - val_accuracy: 0.0862 - val_loss: 5.8788 - learning_rate: 2.5000e-04
```

Epoch 11/20

```
114/114 ████████████████████████████████ 5s 43ms/step - accuracy: 0.2709 - loss: 2.7254 - val_accuracy: 0.1008 - val_loss: 4.0152 - learning_rate: 2.5000e-04
```

Estrategia 2: Transfer Learning

En esta estrategia se toma un modelo ya entrenado con un dataset muy grande y se adapta para que funcione bien en un dataset diferente, más pequeño y específico.

Es decir, en lugar de entrenar un modelo desde cero para una tarea específica (lo cual requeriría muchísimos datos y tiempo de cómputo), se aprovecha el conocimiento que el modelo pre-entrenado ya tiene.

En este caso se ha decidido utilizar los siguientes modelos preentrenados:

1. ResNet50
2. MobileNetV2
3. Xception
4. InceptionResNetV2

Los pasos a seguir en cada modelo son:

1. Cargar un modelo pre-entrenado sin la última capa:
 - Se carga la arquitectura del modelo junto con los pesos aprendidos en el dataset grande (como ImageNet).
 - Se quita la última capa clasificadora original, ya que estaba diseñada para las clases del dataset original.
2. Congelar (no actualizar los pesos durante las primeras fases del entrenamiento) las capas base:
 - Las primeras capas de los modelos grandes aprenden características muy generales y reutilizables. Esto es valioso para casi cualquier tarea similar.
 - Para preservar este conocimiento y evitar que se destruya al empezar a entrenar con nuestros datos, se "congelan" la mayoría de las capas del modelo base.
3. Añadir una nueva capa clasificadora final:
 - Se añaden nuevas capas encima de la base congelada, las cuales serán entrenables.
4. Entrenar la nueva última capa:
 - Se entrena el modelo completo (base congelada + última capa nueva y entrenable) usando las imágenes de Yoga.
 - En esta fase, solo los pesos de la capa de clasificación se ajustan. El modelo aprende a mapear las características

generales extraídas por la base a las clases específicas.

5. Descongelar y fine-tuning:

- Una vez que la nueva capa ha aprendido a clasificar razonablemente bien, se pueden descongelar algunas de las últimas capas de la base pre-entrenada.
- De esta manera se permite que se ajusten ligeramente a los matices de las imágenes de Yoga para mejorar el rendimiento.
- Se vuelve a entrenar el modelo.

```
In [ ]: ### Generamos nuevo conjunto de Datos para ResNet50 y MobileNetV2

import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import os
import pandas as pd
from sklearn.model_selection import train_test_split

# --- 1. Definir rutas ---
base_folder = "DL"
final_folder = os.path.join(base_folder, "final")

img_size = (224, 224)
batch_size = 32

# --- 2. Cargar imágenes y etiquetas ---
images = []
labels = []
for class_folder in os.listdir(final_folder):
    class_path = os.path.join(final_folder, class_folder)
    if os.path.isdir(class_path):
        label = class_folder.split("_")[1] # Extraer etiqueta (ajusta según
        for fname in os.listdir(class_path):
            if fname.lower().endswith(('.jpg', '.jpeg', '.png')):
                images.append(os.path.join(class_path, fname)) # Aquí se as
                labels.append(label)

# --- 3. Crear el Dataframe ---
df_r = pd.DataFrame({'image': images, 'label': labels})

# --- 4. Dividir en train, val y test ---
df_train_r, df_temp_r = train_test_split(df_r, test_size=0.2, stratify=df_r[
df_val_r, df_test_r = train_test_split(df_temp_r, test_size=0.5, stratify=df

# --- 5. Generadores ---
train_datagen_r = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
```



```

        horizontal_flip=True,
        fill_mode='nearest',
    )

val_test_datagen_r = ImageDataGenerator(rescale=1./255)

train_generator_r = train_datagen_r.flow_from_dataframe(
    df_train_r,
    x_col='image',
    y_col='label',
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=True
)

val_generator_r = val_test_datagen_r.flow_from_dataframe(
    df_val_r,
    x_col='image',
    y_col='label',
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False
)

test_generator_r = val_test_datagen_r.flow_from_dataframe(
    df_test_r,
    x_col='image',
    y_col='label',
    target_size=img_size,
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False
)

```

Found 3609 validated image filenames belonging to 47 classes.
 Found 451 validated image filenames belonging to 47 classes.
 Found 452 validated image filenames belonging to 47 classes.

ResNet50

ResNet50 es una red neuronal profunda que pertenece a la familia de redes residuales. Consta de 50 capas. Utiliza bloques residuales para resolver el problema de la degradación de las redes profundas (es decir, el rendimiento que no mejora al agregar más capas).

Cada bloque residual tiene una "conexión de atajo" que permite que la información fluya sin pasar por todas las capas. Esto facilita el entrenamiento de redes muy profundas.

ResNet50 está entrenado en ImageNet, lo que permite que sea utilizado para transfer learning con buenos resultados.

Las ventajas de ResNet50 son las siguientes:

- La red residual resuelve problemas de vanishing gradients (gradientes que se vuelven muy pequeños en redes profundas) y mejora el rendimiento con redes más profundas.
- Es relativamente eficiente en términos de complejidad computacional para una red tan profunda.
- Gracias a los bloques residuales, se mejora la capacidad de generalización en conjuntos de datos complejos.

En cuanto a sus desventajas:

- Aunque eficiente, sigue siendo un modelo relativamente pesado y puede requerir grandes cantidades de memoria y tiempo de procesamiento en comparación con modelos más pequeños como MobileNetV2.
- La red tiene más parámetros debido a la profundidad de la red, lo que puede aumentar el riesgo de sobreajuste si no se utiliza correctamente.

```
In [ ]: from tensorflow.keras.applications import ResNet50
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Dropout, GlobalAveragePooling2D,
from tensorflow.keras.regularizers import l2
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping

# --- 1. Base del modelo ---
base_model_r = ResNet50(include_top=False, weights='imagenet', input_shape=(

# Congelar pesos de la base para transferencia de aprendizaje inicial
base_model_r.trainable = False

# --- 2. Capas superiores personalizadas ---
x_r = base_model_r.output
x_r = GlobalAveragePooling2D()(x_r)
x_r = BatchNormalization()(x_r)
x_r = Dropout(0.5)(x_r)
x_r = Dense(512, activation='relu', kernel_regularizer=l2(0.01))(x_r)
x_r = Dropout(0.5)(x_r)

num_classes_r = len(train_generator_r.class_indices)

predictions_r = Dense(num_classes_r, activation='softmax')(x_r)

model_r = Model(inputs=base_model_r.input, outputs=predictions_r)
```

```

# --- 3. Compilar modelo ---
model_r.compile(
    optimizer=Adam(learning_rate=1e-4),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

# --- 4. Callbacks (opcional pero recomendable) ---
early_stop_r = EarlyStopping(monitor='val_loss', patience=4, restore_best_weights=True)

# --- 5. Entrenamiento ---
history_r = model_r.fit(
    train_generator_r,
    validation_data=val_generator_r,
    epochs=20,
    callbacks=[early_stop_r]
)

# Fine-Tuning (descongelar parte de la base)

base_model_r.trainable = True
# Puedes congelar solo las primeras capas si quieres:
# for layer in base_model.layers[:100]: layer.trainable = False

model_r.compile(optimizer=tf.keras.optimizers.Adam(1e-5), loss='categorical_crossentropy')
history_tun_r=model_r.fit(train_generator_r, validation_data=val_generator_r, epochs=20, callbacks=[early_stop_r])

# Evaluar modelos y comparar precisión
test_loss_r, test_acc_r = model_r.evaluate(test_generator_r)
print("Test accuracy:", test_acc_r)

#Guardamos el test_accuracy en una variable
resnet_acc = test_acc_r

```

Epoch 1/20
113/113 ██ **58s** 413ms/step - accuracy: 0.0246 - loss: 11.6276 - val_accuracy: 0.0288 - val_loss: 9.4019

Epoch 2/20
113/113 ██ **38s** 338ms/step - accuracy: 0.0400 - loss: 9.0617 - val_accuracy: 0.0510 - val_loss: 7.7932

Epoch 3/20
113/113 ██ **38s** 338ms/step - accuracy: 0.0625 - loss: 7.5901 - val_accuracy: 0.1242 - val_loss: 6.8271

Epoch 4/20
113/113 ██ **38s** 337ms/step - accuracy: 0.0638 - loss: 6.8189 - val_accuracy: 0.1685 - val_loss: 6.1868

Epoch 5/20
113/113 ██ **38s** 333ms/step - accuracy: 0.0800 - loss: 6.3401 - val_accuracy: 0.1907 - val_loss: 5.7441

Epoch 6/20
113/113 ██ **38s** 337ms/step - accuracy: 0.0791 - loss: 5.9822 - val_accuracy: 0.1929 - val_loss: 5.4228

Epoch 7/20
113/113 ██ **38s** 337ms/step - accuracy: 0.0884 - loss: 5.7184 - val_accuracy: 0.2018 - val_loss: 5.1755

Epoch 8/20
113/113 ██ **38s** 335ms/step - accuracy: 0.0897 - loss: 5.5575 - val_accuracy: 0.1951 - val_loss: 4.9904

Epoch 9/20
113/113 ██ **38s** 339ms/step - accuracy: 0.0861 - loss: 5.3781 - val_accuracy: 0.2173 - val_loss: 4.8340

Epoch 10/20
113/113 ██ **38s** 337ms/step - accuracy: 0.1024 - loss: 5.2314 - val_accuracy: 0.2195 - val_loss: 4.6951

Epoch 11/20
113/113 ██ **38s** 337ms/step - accuracy: 0.1187 - loss: 5.0516 - val_accuracy: 0.2417 - val_loss: 4.5660

Epoch 12/20
113/113 ██ **38s** 338ms/step - accuracy: 0.1159 - loss: 4.9586 - val_accuracy: 0.2373 - val_loss: 4.4575

Epoch 13/20
113/113 ██ **38s** 336ms/step - accuracy: 0.1243 - loss: 4.8433 - val_accuracy: 0.2506 - val_loss: 4.3526

Epoch 14/20
113/113 ██ **38s** 340ms/step - accuracy: 0.1153 - loss: 4.8029 - val_accuracy: 0.2395 - val_loss: 4.2615

Epoch 15/20
113/113 ██ **38s** 337ms/step - accuracy: 0.1275 - loss: 4.6905 - val_accuracy: 0.2483 - val_loss: 4.1752

Epoch 16/20
113/113 ██ **38s** 338ms/step - accuracy: 0.1366 - loss: 4.6019 - val_accuracy: 0.2528 - val_loss: 4.1036

Epoch 17/20
113/113 ██ **38s** 335ms/step - accuracy: 0.1434 - loss: 4.4713 - val_accuracy: 0.2572 - val_loss: 4.0213

Epoch 18/20
113/113 ██ **38s** 336ms/step - accuracy: 0.1355 - loss: 4.4221 - val_accuracy: 0.2905 - val_loss: 3.9562

Epoch 19/20
113/113 ██ **38s** 338ms/step - accuracy: 0.1417 - loss: 4.3555

- val_accuracy: 0.2639 - val_loss: 3.8810
Epoch 20/20
113/113 ██ **38s** 338ms/step - accuracy: 0.1435 - loss: 4.3093
- val_accuracy: 0.2749 - val_loss: 3.8186
Epoch 1/20
113/113 ██ **118s** 565ms/step - accuracy: 0.0441 - loss: 4.9562
- val_accuracy: 0.0222 - val_loss: 5.3530
Epoch 2/20
113/113 ██ **38s** 338ms/step - accuracy: 0.0869 - loss: 4.4715
- val_accuracy: 0.0310 - val_loss: 5.2285
Epoch 3/20
113/113 ██ **39s** 340ms/step - accuracy: 0.1474 - loss: 4.1757
- val_accuracy: 0.0200 - val_loss: 5.6277
Epoch 4/20
113/113 ██ **38s** 338ms/step - accuracy: 0.2133 - loss: 3.8792
- val_accuracy: 0.0399 - val_loss: 5.0973
Epoch 5/20
113/113 ██ **38s** 339ms/step - accuracy: 0.2777 - loss: 3.6034
- val_accuracy: 0.0820 - val_loss: 4.4788
Epoch 6/20
113/113 ██ **38s** 339ms/step - accuracy: 0.3193 - loss: 3.4078
- val_accuracy: 0.1907 - val_loss: 3.8603
Epoch 7/20
113/113 ██ **38s** 337ms/step - accuracy: 0.4083 - loss: 3.1179
- val_accuracy: 0.3858 - val_loss: 3.2009
Epoch 8/20
113/113 ██ **38s** 339ms/step - accuracy: 0.4559 - loss: 2.9283
- val_accuracy: 0.5654 - val_loss: 2.6911
Epoch 9/20
113/113 ██ **38s** 338ms/step - accuracy: 0.5227 - loss: 2.7230
- val_accuracy: 0.6829 - val_loss: 2.3121
Epoch 10/20
113/113 ██ **38s** 339ms/step - accuracy: 0.5607 - loss: 2.5707
- val_accuracy: 0.7384 - val_loss: 2.0695
Epoch 11/20
113/113 ██ **38s** 340ms/step - accuracy: 0.5927 - loss: 2.3937
- val_accuracy: 0.7738 - val_loss: 1.9568
Epoch 12/20
113/113 ██ **38s** 339ms/step - accuracy: 0.6371 - loss: 2.2765
- val_accuracy: 0.8004 - val_loss: 1.8804
Epoch 13/20
113/113 ██ **39s** 341ms/step - accuracy: 0.6847 - loss: 2.1421
- val_accuracy: 0.7916 - val_loss: 1.7938
Epoch 14/20
113/113 ██ **39s** 342ms/step - accuracy: 0.7166 - loss: 2.0698
- val_accuracy: 0.8093 - val_loss: 1.7116
Epoch 15/20
113/113 ██ **38s** 339ms/step - accuracy: 0.7116 - loss: 1.9813
- val_accuracy: 0.8271 - val_loss: 1.6465
Epoch 16/20
113/113 ██ **39s** 342ms/step - accuracy: 0.7336 - loss: 1.9055
- val_accuracy: 0.8315 - val_loss: 1.5888
Epoch 17/20
113/113 ██ **39s** 342ms/step - accuracy: 0.7704 - loss: 1.8070
- val_accuracy: 0.8381 - val_loss: 1.5461
Epoch 18/20

```
113/113 ████████████████████████████████ 39s 341ms/step - accuracy: 0.7895 - loss: 1.7316
- val_accuracy: 0.8492 - val_loss: 1.5076
Epoch 19/20
113/113 ████████████████████████████████ 39s 341ms/step - accuracy: 0.8027 - loss: 1.6587
- val_accuracy: 0.8492 - val_loss: 1.4719
Epoch 20/20
113/113 ████████████████████████████████ 38s 338ms/step - accuracy: 0.8060 - loss: 1.6403
- val_accuracy: 0.8603 - val_loss: 1.4433
15/15 ████████████████████████████████ 3s 184ms/step - accuracy: 0.8856 - loss: 1.3761
Test accuracy: 0.8761062026023865
```

MobileNetV2

MobileNetV2 es un modelo eficiente diseñado específicamente para dispositivos con recursos limitados. Utiliza bloques de convoluciones invertidas y convoluciones separables en profundidad para mejorar la eficiencia.

Está diseñado para ser rápido y eficiente en dispositivos con poder computacional limitado como teléfonos. MobileNetV2 también está entrenado en ImageNet.

Sus ventajas son:

- Es un modelo muy eficiente en cuanto a la cantidad de operaciones necesarias, lo que lo hace adecuado para su uso en dispositivos móviles y en aplicaciones en tiempo real.
- Tiene un número de parámetros significativamente menor en comparación con modelos más grandes, lo que reduce los requerimientos de memoria y mejora la velocidad de procesamiento.
- A pesar de ser ligero, ofrece un buen rendimiento en tareas de clasificación de imágenes.

En cuanto a sus desventajas:

- Aunque es eficiente, su rendimiento puede no ser tan alto como el de redes más grandes como ResNet50 o Xception, especialmente en tareas complejas o con grandes conjuntos de datos.
- Debido a su enfoque en eficiencia, no es la mejor opción para tareas extremadamente complejas donde se requiere la máxima precisión.

```
In [ ]: import tensorflow as tf
        from tensorflow.keras.applications import MobileNetV2
```

```

from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Dropout, GlobalAveragePooling2D,
from tensorflow.keras.regularizers import l2
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.optimizers import Adam

# Parámetros
img_size = (224, 224)
batch_size = 32
num_epochs = 20

# Número de clases desde los generadores ya creados para ResNet50
num_classes_M = len(train_generator_r.class_indices)

# --- Cargar MobileNetV2 como base ---
base_modelM = MobileNetV2(
    input_shape=(img_size[0], img_size[1], 3),
    include_top=False,
    weights='imagenet'
)

# Congelar la base (solo entrenamiento de capas superiores)
base_modelM.trainable = False

# --- Construir modelo con capas personalizadas ---
x_M = base_modelM.output
x_M = GlobalAveragePooling2D()(x_M)
x_M = BatchNormalization()(x_M)
x_M = Dropout(0.5)(x_M)
x_M = Dense(256, activation='relu', kernel_regularizer=l2(0.01))(x_M)
x_M = Dropout(0.5)(x_M)
predictions_M = Dense(num_classes_M, activation='softmax')(x_M)

modelM = Model(inputs=base_modelM.input, outputs=predictions_M)

# --- Compilar modelo ---
modelM.compile(
    optimizer=Adam(learning_rate=1e-4),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

# --- Callback para early stopping ---
early_stop_M = EarlyStopping(monitor='val_loss', patience=3, restore_best_weights=True)

# --- Entrenar solo capas superiores ---
history_m = modelM.fit(
    train_generator_r,
    validation_data=val_generator_r,
    epochs=num_epochs,
    callbacks=[early_stop_M]
)

# --- Fine-tuning: descongelar base ---
base_modelM.trainable = True
modelM.compile(optimizer=Adam(learning_rate=1e-5), loss='categorical_crossentropy', metrics=['accuracy'])

```

```
# --- Entrenar modelo completo ---
history_tun_M= modelM.fit(
    train_generator_r,
    validation_data=val_generator_r,
    epochs=20,
    callbacks=[early_stop_M]
)

# --- Evaluación final ---
test_loss_M, test_acc_M = modelM.evaluate(test_generator_r)
print(f"Precisión en test con MobileNetV2: {test_acc_M:.4f}")

# Guardamos el test_accuracy en una variable
mobilenet_acc = test_acc_M
```



```

Download data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_weights_tf_dim_ordering_tf_kernels_1.0_224_no_top.h5
9406464/9406464 00000000000000000000 2s 0us/step
Epoch 1/20
113/113 00000000000000000000 74s 514ms/step - accuracy: 0.0223 - loss: 10.0590 - val_accuracy: 0.0776 - val_loss: 7.7636
Epoch 2/20
113/113 00000000000000000000 38s 336ms/step - accuracy: 0.0421 - loss: 9.1906 - val_accuracy: 0.1996 - val_loss: 7.2412
Epoch 3/20
113/113 00000000000000000000 38s 338ms/step - accuracy: 0.0720 - loss: 8.6347 - val_accuracy: 0.2927 - val_loss: 6.7796
Epoch 4/20
113/113 00000000000000000000 39s 340ms/step - accuracy: 0.0939 - loss: 8.0416 - val_accuracy: 0.3681 - val_loss: 6.3875
Epoch 5/20
113/113 00000000000000000000 38s 340ms/step - accuracy: 0.1319 - loss: 7.6895 - val_accuracy: 0.4257 - val_loss: 6.0761
Epoch 6/20
113/113 00000000000000000000 38s 338ms/step - accuracy: 0.1515 - loss: 7.3321 - val_accuracy: 0.4523 - val_loss: 5.8116
Epoch 7/20
113/113 00000000000000000000 39s 341ms/step - accuracy: 0.1928 - loss: 7.0143 - val_accuracy: 0.4967 - val_loss: 5.5848
Epoch 8/20
113/113 00000000000000000000 38s 338ms/step - accuracy: 0.2116 - loss: 6.7142 - val_accuracy: 0.5188 - val_loss: 5.3743
Epoch 9/20
113/113 00000000000000000000 38s 338ms/step - accuracy: 0.2282 - loss: 6.5351 - val_accuracy: 0.5632 - val_loss: 5.1848
Epoch 10/20
113/113 00000000000000000000 38s 338ms/step - accuracy: 0.2540 - loss: 6.2676 - val_accuracy: 0.5787 - val_loss: 5.0029
Epoch 11/20
113/113 00000000000000000000 38s 336ms/step - accuracy: 0.3009 - loss: 5.9901 - val_accuracy: 0.5965 - val_loss: 4.8378
Epoch 12/20
113/113 00000000000000000000 38s 335ms/step - accuracy: 0.3056 - loss: 5.8525 - val_accuracy: 0.6297 - val_loss: 4.6808
Epoch 13/20
113/113 00000000000000000000 38s 339ms/step - accuracy: 0.3078 - loss: 5.6890 - val_accuracy: 0.6497 - val_loss: 4.5405
Epoch 14/20
113/113 00000000000000000000 38s 339ms/step - accuracy: 0.3437 - loss: 5.4946 - val_accuracy: 0.6563 - val_loss: 4.4098
Epoch 15/20
113/113 00000000000000000000 38s 339ms/step - accuracy: 0.3619 - loss: 5.4122 - val_accuracy: 0.6696 - val_loss: 4.2813
Epoch 16/20
113/113 00000000000000000000 38s 336ms/step - accuracy: 0.3815 - loss: 5.1680 - val_accuracy: 0.6851 - val_loss: 4.1587
Epoch 17/20
113/113 00000000000000000000 38s 337ms/step - accuracy: 0.4000 - loss: 5.0230 - val_accuracy: 0.6874 - val_loss: 4.0480
Epoch 18/20

```

113/113 00000000000000000000 38s 335ms/step - accuracy: 0.3868 - loss: 4.9643
- val_accuracy: 0.6962 - val_loss: 3.9368
Epoch 19/20
113/113 00000000000000000000 38s 336ms/step - accuracy: 0.4172 - loss: 4.7504
- val_accuracy: 0.7029 - val_loss: 3.8358
Epoch 20/20
113/113 00000000000000000000 38s 335ms/step - accuracy: 0.4489 - loss: 4.5748
- val_accuracy: 0.7029 - val_loss: 3.7411
Epoch 1/20
113/113 00000000000000000000 121s 606ms/step - accuracy: 0.1870 - loss: 5.827
2 - val_accuracy: 0.6718 - val_loss: 3.7253
Epoch 2/20
113/113 00000000000000000000 38s 338ms/step - accuracy: 0.2205 - loss: 5.5795
- val_accuracy: 0.6297 - val_loss: 3.7356
Epoch 3/20
113/113 00000000000000000000 38s 338ms/step - accuracy: 0.2597 - loss: 5.3361
- val_accuracy: 0.6253 - val_loss: 3.7445
Epoch 4/20
113/113 00000000000000000000 38s 336ms/step - accuracy: 0.2879 - loss: 5.1895
- val_accuracy: 0.6297 - val_loss: 3.7149
Epoch 5/20
113/113 00000000000000000000 38s 334ms/step - accuracy: 0.3121 - loss: 5.0743
- val_accuracy: 0.6563 - val_loss: 3.6914
Epoch 6/20
113/113 00000000000000000000 38s 340ms/step - accuracy: 0.3614 - loss: 4.8702
- val_accuracy: 0.6608 - val_loss: 3.6625
Epoch 7/20
113/113 00000000000000000000 38s 336ms/step - accuracy: 0.3726 - loss: 4.7772
- val_accuracy: 0.6585 - val_loss: 3.6246
Epoch 8/20
113/113 00000000000000000000 38s 335ms/step - accuracy: 0.4141 - loss: 4.6620
- val_accuracy: 0.6652 - val_loss: 3.5942
Epoch 9/20
113/113 00000000000000000000 38s 339ms/step - accuracy: 0.4117 - loss: 4.6208
- val_accuracy: 0.6763 - val_loss: 3.5590
Epoch 10/20
113/113 00000000000000000000 38s 338ms/step - accuracy: 0.4219 - loss: 4.5347
- val_accuracy: 0.6829 - val_loss: 3.5322
Epoch 11/20
113/113 00000000000000000000 38s 338ms/step - accuracy: 0.4525 - loss: 4.4028
- val_accuracy: 0.6984 - val_loss: 3.4984
Epoch 12/20
113/113 00000000000000000000 38s 338ms/step - accuracy: 0.4731 - loss: 4.2969
- val_accuracy: 0.7007 - val_loss: 3.4587
Epoch 13/20
113/113 00000000000000000000 38s 339ms/step - accuracy: 0.4970 - loss: 4.2497
- val_accuracy: 0.7007 - val_loss: 3.4325
Epoch 14/20
113/113 00000000000000000000 38s 337ms/step - accuracy: 0.4988 - loss: 4.2399
- val_accuracy: 0.7184 - val_loss: 3.3899
Epoch 15/20
113/113 00000000000000000000 38s 338ms/step - accuracy: 0.5101 - loss: 4.1625
- val_accuracy: 0.7228 - val_loss: 3.3586
Epoch 16/20
113/113 00000000000000000000 38s 336ms/step - accuracy: 0.5315 - loss: 4.0324
- val_accuracy: 0.7450 - val_loss: 3.3159

Epoch 17/20

113/113 ████████████████████████████████ **38s** 335ms/step - accuracy: 0.5225 - loss: 4.0495
- val_accuracy: 0.7494 - val_loss: 3.2759

Epoch 18/20

113/113 ████████████████████████████████ **38s** 337ms/step - accuracy: 0.5470 - loss: 3.9849
- val_accuracy: 0.7583 - val_loss: 3.2528

Epoch 19/20

113/113 ████████████████████████████████ **38s** 337ms/step - accuracy: 0.5579 - loss: 3.9498
- val_accuracy: 0.7672 - val_loss: 3.2174

Epoch 20/20

113/113 ████████████████████████████████ **38s** 339ms/step - accuracy: 0.5652 - loss: 3.9235
- val_accuracy: 0.7694 - val_loss: 3.1898

15/15 ████████████████████████████████ **5s** 353ms/step - accuracy: 0.7810 - loss: 3.0822

Precisión en test con MobileNetV2: 0.7810

Xception

Para este modelo primero se genera un nuevo conjunto de datos, ya que tiene una configuración de tamaño de imagen diferente: 299 * 299, en lugar de 224 * 224 como usa ResNet o MobileNet.

```
In [ ]: ### Generamos nuevo conjunto de Datos para XCEPTION

import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import os
import pandas as pd
from sklearn.model_selection import train_test_split

# --- 1. Definir rutas ---
base_folder = "DL"
final_folder = os.path.join(base_folder, "final")

img_size_x = (299, 299) # Para usar Inception o Xception
batch_size_x = 16

# --- 2. Cargar imágenes y etiquetas ---
images = []
labels = []
for class_folder in os.listdir(final_folder):
    class_path = os.path.join(final_folder, class_folder)
    if os.path.isdir(class_path):
        label = class_folder.split("_")[1] # Extraer etiqueta (ajusta según
        for fname in os.listdir(class_path):
            if fname.lower().endswith(('.jpg', '.jpeg', '.png')):
                images.append(os.path.join(class_path, fname)) # Aquí se as
                labels.append(label)

# --- 3. Crear el Dataframe ---
df29 = pd.DataFrame({'image': images, 'label': labels})

# --- 4. Dividir en train, val y test ---
df_train_x, df_temp_x = train_test_split(df29, test_size=0.2, stratify=df29[
df_val_x, df_test_x = train_test_split(df_temp_x, test_size=0.5, stratify=df
```

```

# --- 5. Generadores ---
train_datagen29 = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

val_test_datagen29 = ImageDataGenerator(rescale=1./255)

train_generator29 = train_datagen29.flow_from_dataframe(
    df_train_x,
    x_col='image',
    y_col='label',
    target_size=img_size_x,
    batch_size=batch_size_x,
    class_mode='categorical',
    shuffle=True
)

val_generator29 = val_test_datagen29.flow_from_dataframe(
    df_val_x,
    x_col='image',
    y_col='label',
    target_size=img_size_x,
    batch_size=batch_size_x,
    class_mode='categorical',
    shuffle=False
)

test_generator29 = val_test_datagen29.flow_from_dataframe(
    df_test_x,
    x_col='image',
    y_col='label',
    target_size=img_size_x,
    batch_size=batch_size_x,
    class_mode='categorical',
    shuffle=False
)

```

Found 3609 validated image filenames belonging to 47 classes.

Found 451 validated image filenames belonging to 47 classes.

Found 452 validated image filenames belonging to 47 classes.

Xception es una variante de Inception, pero en lugar de utilizar convoluciones estándar, utiliza convoluciones separables en profundidad. Este tipo de convolución separa las operaciones de convolución en dos pasos: una convolución espacial y una convolución a lo largo de los canales.

Está diseñado para ser un modelo de alto rendimiento con menos parámetros que InceptionV3, y puede superar a modelos como ResNet en tareas de clasificación de imágenes. Este también está preentrenado en ImageNet.

Sus ventajas son:

- Las convoluciones separables en profundidad permiten reducir la cantidad de parámetros sin sacrificar el rendimiento.
- Xception es más eficiente computacionalmente que otros modelos tradicionales como VGG, y ofrece un rendimiento excelente en comparación con modelos más simples.
- Xception muestra una excelente capacidad para generalizar en diferentes tipos de datos de imágenes, lo que lo hace adecuado para tareas complejas.

Respecto a sus desventajas:

- Aunque más eficiente que otros modelos como VGG, sigue siendo un modelo grande que requiere más recursos de cómputo y memoria en comparación con redes ligeras como MobileNetV2.
- Su arquitectura, basada en convoluciones separables, puede ser más compleja de entender y ajustar para nuevos usuarios.

```
In [ ]: import tensorflow as tf
from tensorflow.keras.applications import Xception
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Dropout, GlobalAveragePooling2D,
from tensorflow.keras.regularizers import l2
from tensorflow.keras.callbacks import EarlyStopping

# Parámetros
img_size_x = (299, 299)
batch_size_x = 16

# Modelo base con pesos de ImageNet
base_model1 = Xception(input_shape=(img_size_x[0], img_size_x[1], 3), include_top=False, weights='imagenet', trainable=False) # Congelado

# Capas superiores personalizadas
x_x = base_model1.output
x_x = GlobalAveragePooling2D()(x_x)
x_x = BatchNormalization()(x_x)
x_x = Dropout(0.5)(x_x)
x_x = Dense(512, activation='relu', kernel_regularizer=l2(0.01))(x_x)
x_x = Dropout(0.5)(x_x)
```

```

# Número de clases desde los generadores
num_classes_x = len(train_generator29.class_indices)

predictions_x = Dense(num_classes_x, activation='softmax')(x_x)

# Construir el modelo
model1 = Model(inputs=base_model1.input, outputs=predictions_x)

# Compilar el modelo
model1.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['

# Early stopping para evitar sobreentrenamiento
early_stop_x = EarlyStopping(patience=3, restore_best_weights=True)

# Entrenamiento inicial (solo capas superiores)
model1.fit(train_generator29, validation_data=val_generator29, epochs=7, cal

# Descongelar el modelo base para fine-tuning
base_model1.trainable = True

# Recompilar el modelo con tasa de aprendizaje más baja
model1.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=1e-5),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

# Entrenar todo el modelo (fine-tuning)
history_x= model1.fit(
    train_generator29,
    validation_data=val_generator29,
    epochs=16, # Ajusta según los resultados obtenidos en la primera fase d
    callbacks=[early_stop_x]
)

# Evaluar en el conjunto de test
test_loss_x, test_acc_x = model1.evaluate(test_generator29)
print(f"Precisión en test con Xception (fine-tuning): {test_acc_x:.4f}")

#Guardamos el test_accuracy en una variable
xception_acc = test_acc_x

```

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/xception/xception_weights_tf_dim_ordering_tf_kernels_notop.h5

83683744/83683744 ██████████ 4s 0us/step

```

/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_d
ataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super
().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`,
`use_multiprocessing`, `max_queue_size`. Do not pass these arguments to `fit
()`, as they will be ignored.
  self.warn_if_super_not_called()

```

Epoch 1/7
226/226 00000000000000000000 101s 403ms/step - accuracy: 0.1118 - loss: 12.48
19 - val_accuracy: 0.5521 - val_loss: 8.6408
Epoch 2/7
226/226 00000000000000000000 66s 292ms/step - accuracy: 0.3787 - loss: 8.7685
- val_accuracy: 0.6098 - val_loss: 6.5769
Epoch 3/7
226/226 00000000000000000000 66s 294ms/step - accuracy: 0.4410 - loss: 7.0131
- val_accuracy: 0.6452 - val_loss: 5.3059
Epoch 4/7
226/226 00000000000000000000 66s 291ms/step - accuracy: 0.5024 - loss: 5.8148
- val_accuracy: 0.6763 - val_loss: 4.5589
Epoch 5/7
226/226 00000000000000000000 66s 291ms/step - accuracy: 0.5049 - loss: 5.1113
- val_accuracy: 0.6630 - val_loss: 4.2395
Epoch 6/7
226/226 00000000000000000000 66s 291ms/step - accuracy: 0.5233 - loss: 4.7336
- val_accuracy: 0.6918 - val_loss: 3.9291
Epoch 7/7
226/226 00000000000000000000 66s 291ms/step - accuracy: 0.5331 - loss: 4.4138
- val_accuracy: 0.6785 - val_loss: 3.8711
Epoch 1/16
226/226 00000000000000000000 133s 413ms/step - accuracy: 0.3954 - loss: 4.970
2 - val_accuracy: 0.6718 - val_loss: 3.8099
Epoch 2/16
226/226 00000000000000000000 66s 292ms/step - accuracy: 0.5094 - loss: 4.4377
- val_accuracy: 0.7206 - val_loss: 3.6347
Epoch 3/16
226/226 00000000000000000000 66s 291ms/step - accuracy: 0.5596 - loss: 4.2165
- val_accuracy: 0.7605 - val_loss: 3.5175
Epoch 4/16
226/226 00000000000000000000 66s 291ms/step - accuracy: 0.6180 - loss: 3.9800
- val_accuracy: 0.7761 - val_loss: 3.4172
Epoch 5/16
226/226 00000000000000000000 66s 291ms/step - accuracy: 0.6537 - loss: 3.8148
- val_accuracy: 0.8004 - val_loss: 3.3314
Epoch 6/16
226/226 00000000000000000000 67s 295ms/step - accuracy: 0.6650 - loss: 3.7399
- val_accuracy: 0.8137 - val_loss: 3.2543
Epoch 7/16
226/226 00000000000000000000 67s 295ms/step - accuracy: 0.7018 - loss: 3.5692
- val_accuracy: 0.8337 - val_loss: 3.1895
Epoch 8/16
226/226 00000000000000000000 66s 291ms/step - accuracy: 0.7514 - loss: 3.4186
- val_accuracy: 0.8448 - val_loss: 3.1339
Epoch 9/16
226/226 00000000000000000000 66s 294ms/step - accuracy: 0.7435 - loss: 3.4156
- val_accuracy: 0.8404 - val_loss: 3.0828
Epoch 10/16
226/226 00000000000000000000 66s 292ms/step - accuracy: 0.7454 - loss: 3.3967
- val_accuracy: 0.8426 - val_loss: 3.0339
Epoch 11/16
226/226 00000000000000000000 67s 295ms/step - accuracy: 0.7791 - loss: 3.2678
- val_accuracy: 0.8492 - val_loss: 2.9882
Epoch 12/16
226/226 00000000000000000000 67s 298ms/step - accuracy: 0.7804 - loss: 3.2133

```
- val_accuracy: 0.8625 - val_loss: 2.9450
Epoch 13/16
226/226 ████████████████████████████████████ 67s 296ms/step - accuracy: 0.8121 - loss: 3.0852
- val_accuracy: 0.8670 - val_loss: 2.9032
Epoch 14/16
226/226 ████████████████████████████████████ 67s 297ms/step - accuracy: 0.8183 - loss: 3.0513
- val_accuracy: 0.8825 - val_loss: 2.8581
Epoch 15/16
226/226 ████████████████████████████████████ 67s 298ms/step - accuracy: 0.8213 - loss: 3.0236
- val_accuracy: 0.8780 - val_loss: 2.8180
Epoch 16/16
226/226 ████████████████████████████████████ 67s 296ms/step - accuracy: 0.8354 - loss: 2.9358
- val_accuracy: 0.8847 - val_loss: 2.7843
29/29 ████████████████████████████████████ 3s 117ms/step - accuracy: 0.8690 - loss: 2.7594
Precisión en test con Xception (fine-tuning): 0.8827
```

InceptionResNetV2

InceptionResNetV2 combina la arquitectura Inception con conexiones residuales. Esta combinación busca mejorar la capacidad de las redes profundas, a la vez que mantiene un buen equilibrio entre la precisión y la eficiencia computacional. De nuevo, InceptionResNetV2 está preentrenado en ImageNet.

InceptionResNetV2 emplea una combinación de múltiples técnicas de optimización y arquitectura para lograr una red más profunda y eficiente.

Ventajas:

- La combinación de Inception y ResNet mejora la capacidad del modelo para aprender características más complejas y específicas de los datos.
- Debido a su capacidad para combinar diferentes tipos de convoluciones, InceptionResNetV2 es muy flexible y adecuado para una amplia variedad de tareas de visión por computadora.
- En términos generales, InceptionResNetV2 ofrece una excelente precisión para tareas de clasificación de imágenes complejas.

Desventajas:

- Es un modelo grande, lo que puede hacer que su entrenamiento sea más lento y que requiera más recursos computacionales (memoria y poder de procesamiento).
- La combinación de Inception y ResNet hace que la arquitectura sea más difícil de ajustar y de implementar en dispositivos con

recursos limitados.

```
In [ ]: import tensorflow as tf
from tensorflow.keras.applications import InceptionResNetV2
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Dropout, GlobalAveragePooling2D,
from tensorflow.keras.regularizers import l2
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import EarlyStopping

# --- 1. Definir rutas ---
base_folder = "DL"
final_folder = os.path.join(base_folder, "final")

# Parámetros
img_size_i = (299, 299) # InceptionResNetV2 requiere tamaño 299x299
batch_size_i = 16
num_epochs_i = 20

# --- 2. Cargar imágenes y etiquetas ---
images = []
labels = []
for class_folder in os.listdir(final_folder):
    class_path = os.path.join(final_folder, class_folder)
    if os.path.isdir(class_path):
        label = class_folder.split("_")[1] # Extraer etiqueta (ajusta según
        for fname in os.listdir(class_path):
            if fname.lower().endswith(('.jpg', '.jpeg', '.png')):
                images.append(os.path.join(class_path, fname)) # Aquí se as
                labels.append(label)

# --- 3. Crear el Dataframe ---
df_i = pd.DataFrame({'image': images, 'label': labels})

# --- 4. Dividir en train, val y test ---
df_train_i, df_temp_i = train_test_split(df_i, test_size=0.2, stratify=df_i['label'])
df_val_i, df_test_i = train_test_split(df_temp_i, test_size=0.5, stratify=df_temp_i['label'])

# --- Generadores de datos con data augmentation ---
train_datagen_i = ImageDataGenerator(
    rescale=1./255,
    rotation_range=30,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

val_test_datagen_i = ImageDataGenerator(rescale=1./255)

# Cargar generadores de datos
```

```

train_generator_i= train_datagen_i.flow_from_dataframe(
    df_train_i,
    x_col='image',
    y_col='label',
    target_size=img_size_i,
    batch_size=batch_size_i,
    class_mode='categorical',
    shuffle=True
)

val_generator_i = val_test_datagen_i.flow_from_dataframe(
    df_val_i,
    x_col='image',
    y_col='label',
    target_size=img_size_i,
    batch_size=batch_size_i,
    class_mode='categorical',
    shuffle=False
)

test_generator_i = val_test_datagen_i.flow_from_dataframe(
    df_test_i,
    x_col='image',
    y_col='label',
    target_size=img_size_i,
    batch_size=batch_size_i,
    class_mode='categorical',
    shuffle=False
)

# --- Cargar el modelo base preentrenado ---
base_model_i = InceptionResNetV2(input_shape=(img_size_i[0], img_size_i[1],

# Congelar el modelo base
base_model_i.trainable = False

# --- Capas superiores personalizadas ---
x_i = base_model_i.output
x_i = GlobalAveragePooling2D()(x_i)
x_i = BatchNormalization()(x_i)
x_i = Dropout(0.5)(x_i)
x_i = Dense(512, activation='relu', kernel_regularizer=l2(0.01))(x_i)
x_i = Dropout(0.5)(x_i)

# Obtener número de clases
num_classes_i = len(train_generator_i.class_indices)

# Capa de salida con softmax
predictions_i = Dense(num_classes_i, activation='softmax')(x_i)

# --- Construir el modelo ---
model_i = Model(inputs=base_model_i.input, outputs=predictions_i)

# --- Compilar modelo ---
model_i.compile(optimizer='adam', loss='categorical_crossentropy', metrics=[

```

```

# --- Early stopping para evitar sobreentrenamiento ---
early_stop_i = EarlyStopping(patience=3, restore_best_weights=True)

# --- Entrenamiento inicial (solo capas superiores) ---
model_i.fit(train_generator_i, validation_data=val_generator_i, epochs=num_e

# --- Fine-tuning: Descongelar la base ---
base_model_i.trainable = True
model_i.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=1e-5),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

# --- Entrenamiento con fine-tuning (todo el modelo) ---
history_i=model_i.fit(
    train_generator_i,
    validation_data=val_generator_i,
    epochs=num_epochs_i,
    callbacks=[early_stop_i]
)

# --- Evaluación en el conjunto de test ---
test_loss_i, test_acc_i = model_i.evaluate(test_generator_i)
print(f"Precisión en test con InceptionResNetV2: {test_acc_i:.4f}")

#Guardamos el test_accuracy en una variable
inceptionresnet_acc = test_acc_i

```

Found 3609 validated image filenames belonging to 47 classes.

Found 451 validated image filenames belonging to 47 classes.

Found 452 validated image filenames belonging to 47 classes.

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/inception_resnet_v2/inception_resnet_v2_weights_tf_dim_ordering_tf_kernels_notop.h5

219055592/219055592  **10s** 0us/step

```

/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as they will be ignored.
  self._warn_if_super_not_called()

```

Epoch 1/20
226/226 00000000000000000000 **125s** 422ms/step - accuracy: 0.0982 - loss: 12.2775 - val_accuracy: 0.3792 - val_loss: 8.2166

Epoch 2/20
226/226 00000000000000000000 **67s** 295ms/step - accuracy: 0.2734 - loss: 8.7287 - val_accuracy: 0.4723 - val_loss: 6.4382

Epoch 3/20
226/226 00000000000000000000 **67s** 295ms/step - accuracy: 0.3213 - loss: 7.0066 - val_accuracy: 0.5322 - val_loss: 5.1663

Epoch 4/20
226/226 00000000000000000000 **67s** 297ms/step - accuracy: 0.3864 - loss: 5.6520 - val_accuracy: 0.5654 - val_loss: 4.4031

Epoch 5/20
226/226 00000000000000000000 **67s** 298ms/step - accuracy: 0.4177 - loss: 4.9514 - val_accuracy: 0.5455 - val_loss: 4.0215

Epoch 6/20
226/226 00000000000000000000 **67s** 296ms/step - accuracy: 0.4071 - loss: 4.5869 - val_accuracy: 0.5144 - val_loss: 3.8508

Epoch 7/20
226/226 00000000000000000000 **67s** 297ms/step - accuracy: 0.4331 - loss: 4.2911 - val_accuracy: 0.5277 - val_loss: 3.6406

Epoch 8/20
226/226 00000000000000000000 **67s** 295ms/step - accuracy: 0.4114 - loss: 4.2116 - val_accuracy: 0.6120 - val_loss: 3.4193

Epoch 9/20
226/226 00000000000000000000 **66s** 294ms/step - accuracy: 0.4177 - loss: 4.1029 - val_accuracy: 0.5499 - val_loss: 3.4962

Epoch 10/20
226/226 00000000000000000000 **67s** 294ms/step - accuracy: 0.4228 - loss: 4.0126 - val_accuracy: 0.5610 - val_loss: 3.4722

Epoch 11/20
226/226 00000000000000000000 **66s** 293ms/step - accuracy: 0.4176 - loss: 4.0541 - val_accuracy: 0.6142 - val_loss: 3.4262

Epoch 1/20
226/226 00000000000000000000 **284s** 608ms/step - accuracy: 0.3399 - loss: 4.5283 - val_accuracy: 0.6741 - val_loss: 3.1362

Epoch 2/20
226/226 00000000000000000000 **67s** 295ms/step - accuracy: 0.5080 - loss: 3.7444 - val_accuracy: 0.7295 - val_loss: 2.9130

Epoch 3/20
226/226 00000000000000000000 **67s** 296ms/step - accuracy: 0.5774 - loss: 3.4323 - val_accuracy: 0.7916 - val_loss: 2.7486

Epoch 4/20
226/226 00000000000000000000 **67s** 295ms/step - accuracy: 0.6243 - loss: 3.2395 - val_accuracy: 0.8226 - val_loss: 2.6334

Epoch 5/20
226/226 00000000000000000000 **67s** 296ms/step - accuracy: 0.6898 - loss: 3.0323 - val_accuracy: 0.8337 - val_loss: 2.5483

Epoch 6/20
226/226 00000000000000000000 **67s** 295ms/step - accuracy: 0.7130 - loss: 2.8900 - val_accuracy: 0.8426 - val_loss: 2.4724

Epoch 7/20
226/226 00000000000000000000 **67s** 297ms/step - accuracy: 0.7434 - loss: 2.7782 - val_accuracy: 0.8714 - val_loss: 2.4020

Epoch 8/20
226/226 00000000000000000000 **67s** 296ms/step - accuracy: 0.7650 - loss: 2.6829

```

- val_accuracy: 0.8647 - val_loss: 2.3561
Epoch 9/20
226/226 ████████████████████████████████ 67s 298ms/step - accuracy: 0.7984 - loss: 2.6142
- val_accuracy: 0.8758 - val_loss: 2.3106
Epoch 10/20
226/226 ████████████████████████████████ 68s 299ms/step - accuracy: 0.7844 - loss: 2.5652
- val_accuracy: 0.8958 - val_loss: 2.2593
Epoch 11/20
226/226 ████████████████████████████████ 67s 295ms/step - accuracy: 0.8090 - loss: 2.4992
- val_accuracy: 0.8958 - val_loss: 2.2090
Epoch 12/20
226/226 ████████████████████████████████ 67s 297ms/step - accuracy: 0.8478 - loss: 2.3772
- val_accuracy: 0.9091 - val_loss: 2.1667
Epoch 13/20
226/226 ████████████████████████████████ 67s 296ms/step - accuracy: 0.8439 - loss: 2.3447
- val_accuracy: 0.9135 - val_loss: 2.1372
Epoch 14/20
226/226 ████████████████████████████████ 67s 297ms/step - accuracy: 0.8791 - loss: 2.2470
- val_accuracy: 0.9180 - val_loss: 2.1001
Epoch 15/20
226/226 ████████████████████████████████ 67s 297ms/step - accuracy: 0.8703 - loss: 2.2354
- val_accuracy: 0.9157 - val_loss: 2.0667
Epoch 16/20
226/226 ████████████████████████████████ 67s 296ms/step - accuracy: 0.8630 - loss: 2.1849
- val_accuracy: 0.9157 - val_loss: 2.0306
Epoch 17/20
226/226 ████████████████████████████████ 67s 296ms/step - accuracy: 0.9024 - loss: 2.0822
- val_accuracy: 0.9313 - val_loss: 1.9927
Epoch 18/20
226/226 ████████████████████████████████ 66s 294ms/step - accuracy: 0.9045 - loss: 2.0489
- val_accuracy: 0.9224 - val_loss: 1.9637
Epoch 19/20
226/226 ████████████████████████████████ 67s 294ms/step - accuracy: 0.9181 - loss: 1.9967
- val_accuracy: 0.9202 - val_loss: 1.9339
Epoch 20/20
226/226 ████████████████████████████████ 67s 296ms/step - accuracy: 0.9151 - loss: 1.9676
- val_accuracy: 0.9313 - val_loss: 1.8881
29/29 ████████████████████████████████ 8s 265ms/step - accuracy: 0.9220 - loss: 1.8616
Precisión en test con InceptionResNetV2: 0.9292

```

Evaluación y comparación de las dos estrategias analizadas

```

In [ ]: from google.colab import drive
drive.mount('/content/drive')

# Guardar modelo en Google Drive
model1.save("/content/drive/MyDrive/model1.keras")
model_r.save("/content/drive/MyDrive/modelr.keras")
model_i.save("/content/drive/MyDrive/modeli.keras")
modelM.save("/content/drive/MyDrive/modelM.keras")
model.save("/content/drive/MyDrive/model.keras")

```

Mounted at /content/drive

```
In [ ]: #Cargar los modelos
model_cargado = tf.keras.models.load_model("model.keras")
model1_cargado = tf.keras.models.load_model("model1.keras")
modelr_cargado = tf.keras.models.load_model("modelr.keras")
modeli_cargado = tf.keras.models.load_model("modeli.keras")
modelm_cargado = tf.keras.models.load_model("modelm.keras")
```

Evaluación

Una vez entrenados los modelos, a continuación se evalúa el rendimiento de los modelos utilizando el conjunto de validación.

Para ello, se analiza tanto la precisión (accuracy) como la pérdida (loss) de cada modelo durante las distintas épocas, observando su evolución en gráficos comparativos entre entrenamiento y validación.

```
In [ ]: #Defino todos los modelos
model_list = [model, model_r, modelM, model1, model_i] # Lista con los mode
#Modelo 1 usa Val_generator
test_generators = [None, test_generator_r, test_generator_r, test_generator2
model_names = ["Red neuronal 0", "ResNet50", "MobileNetV2", "Xception", "Inc
```

```
In [ ]: fig, axes = plt.subplots(5, 2, figsize=(12,12))

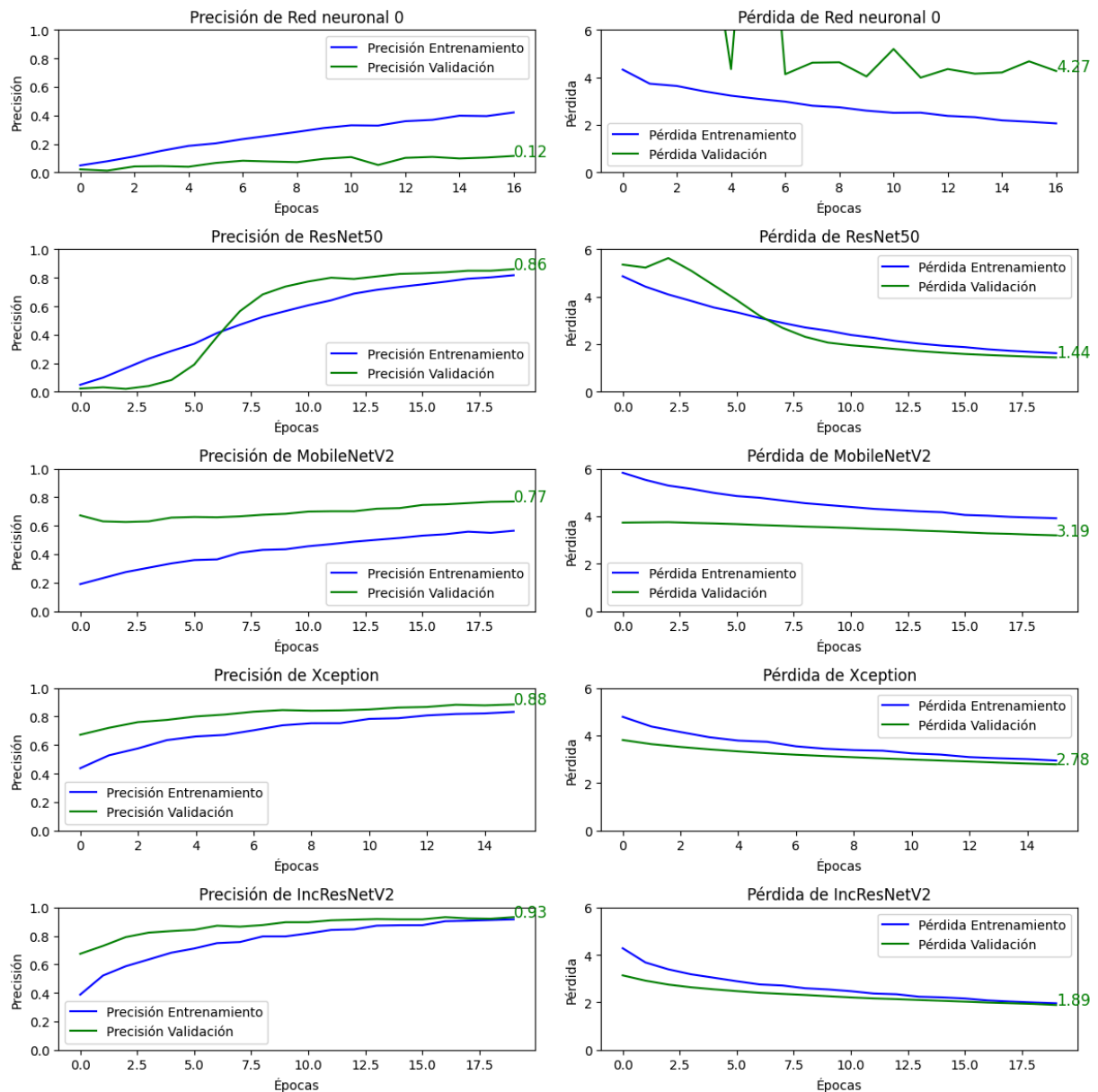
histories = [history, history_tun_r, history_tun_M, history_x, history_i] #
for i in range(5):
    # Precisión
    axes[i,0].plot(histories[i].history['accuracy'], label='Precisión Entren
    axes[i,0].plot(histories[i].history['val_accuracy'], label='Precisión Va
    axes[i,0].set_title(f"Precisión de {model_names[i]}")
    axes[i,0].set_xlabel('Épocas')
    axes[i,0].set_ylabel('Precisión')
    axes[i,0].legend()
    axes[i,0].set_ylim(0, 1) # Rango de precisión de 0 a 1

    # Mostrar el último punto de precisión
    axes[i,0].text(len(histories[i].history['val_accuracy'])-1, histories[i]

    # Pérdida
    axes[i,1].plot(histories[i].history['loss'], label='Pérdida Entrenamient
    axes[i,1].plot(histories[i].history['val_loss'], label='Pérdida Validaci
    axes[i,1].set_title(f"Pérdida de {model_names[i]}")
    axes[i,1].set_xlabel('Épocas')
    axes[i,1].set_ylabel('Pérdida')
    axes[i,1].legend()
    axes[i,1].set_ylim(0, 6) # Rango de pérdida de 0 a 6

    # Mostrar el último punto de precisión
    axes[i,1].text(len(histories[i].history['val_loss'])-1, histories[i].his
```

```
plt.tight_layout()
plt.show()
```



Matriz de confusión y métricas

A continuación, se presenta la matriz de confusión y métricas de cada uno de los modelos estudiados.

```
In [ ]: import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, classification_report
import pandas as pd

reports_dict={}

for i, model in enumerate(model_list):
```

```

print(f"\nEvaluando {model_names[i]}...\n")

if i == 0: # Modelo 1 usa val_generator
    val_images, val_labels = next(val_generator)
    y_true = val_labels
    y_pred = np.argmax(model.predict(val_images), axis=1) # Convertir p
    class_labels = list(val_generator.class_indices.keys()) # Obtener n

else: # Para los otros modelos con test_generator
    y_true = test_generators[i].classes # Clases reales del conjunto de
    y_pred = np.argmax(model.predict(test_generators[i]), axis=1) # Pre
    class_labels = list(test_generators[i].class_indices.keys()) # Etiq

# Filtrar las clases realmente presentes
clases_presentes = np.unique(y_true).astype(int)
filtered_class_labels = [class_labels[i] for i in clases_presentes]

# Genero el informe de clasificación
report_dict = classification_report(y_true, y_pred, labels=clases_presentes)

# Guardo el informe en el diccionario
reports_dict[model_names[i]] = report_dict

# Calcular la matriz de confusión
conf_matrix = confusion_matrix(y_true, y_pred, labels=clases_presentes)

# Visualizar la matriz de confusión
plt.figure(figsize=(10,8))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="coolwarm", linewidths=1)
plt.xlabel("Predicción")
plt.ylabel("Etiqueta Real")
plt.title(f"Matriz de Confusión - {model_names[i]}")
plt.show()

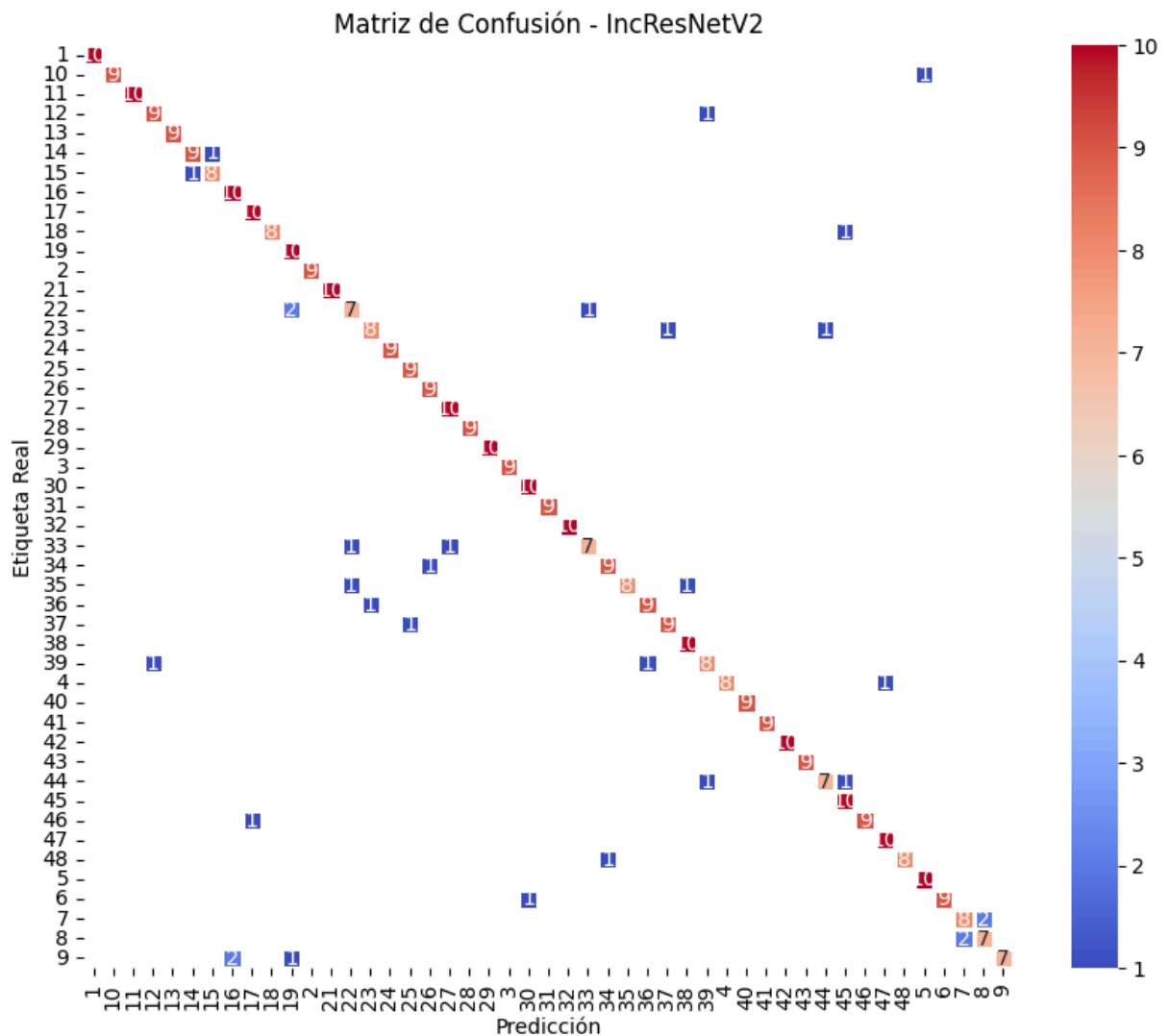
```

Evaluando Red neuronal 0...

1/1 ████████████████████████████████ 0s 71ms/step

[illegible]

15/15 1s 34ms/step



Ahora presentamos la comparación de las métricas de los modelos, sirviendo el color azul para indicar aquellas clases con un mayor valor de precisión, recall y F1-Score.

```
In [ ]: df_reports = []
for model_name, report_dict in reports_dict.items():
    df_report = pd.DataFrame(report_dict).transpose() # Convertir el diccio
    df_report["Modelo"] = model_name # Agregar columna con el nombre del mo
    df_reports.append(df_report)

df_comparacion = pd.concat(df_reports, axis=0).reset_index().rename(columns=

display(df_comparacion.style.set_caption("Comparación de modelos")
        .set_properties(**{'border': '1px solid black'})
        .background_gradient(subset=['precision', 'recall', 'f1-s
```

Comparación de modelos

	Métrica	precision	recall	f1-score	support	Modelo
0	class_11	1.000000	1.000000	1.000000	1.000000	Red neuronal 0
1	class_13	1.000000	1.000000	1.000000	2.000000	Red neuronal 0
2	class_14	1.000000	1.000000	1.000000	2.000000	Red neuronal 0
3	class_15	1.000000	1.000000	1.000000	1.000000	Red neuronal 0
4	class_16	1.000000	1.000000	1.000000	1.000000	Red neuronal 0
5	class_17	1.000000	1.000000	1.000000	1.000000	Red neuronal 0
6	class_18	1.000000	1.000000	1.000000	1.000000	Red neuronal 0
7	class_2	1.000000	1.000000	1.000000	1.000000	Red neuronal 0
8	class_21	1.000000	1.000000	1.000000	1.000000	Red neuronal 0
9	class_25	1.000000	1.000000	1.000000	1.000000	Red neuronal 0
10	class_26	1.000000	1.000000	1.000000	1.000000	Red neuronal 0
11	class_28	1.000000	1.000000	1.000000	1.000000	Red neuronal 0
12	class_3	1.000000	1.000000	1.000000	1.000000	Red neuronal 0
13	class_31	1.000000	1.000000	1.000000	1.000000	Red neuronal 0
14	class_32	1.000000	1.000000	1.000000	1.000000	Red neuronal 0
15	class_33	1.000000	0.500000	0.666667	2.000000	Red neuronal 0
16	class_34	1.000000	1.000000	1.000000	1.000000	Red neuronal 0
17	class_35	1.000000	0.500000	0.666667	2.000000	Red neuronal 0
18	class_37	1.000000	1.000000	1.000000	1.000000	Red neuronal 0
19	class_41	1.000000	1.000000	1.000000	1.000000	Red neuronal 0

	Métrica	precision	recall	f1-score	support	Modelo
20	class_42	1.000000	1.000000	1.000000	1.000000	Red neuronal 0
21	class_45	1.000000	1.000000	1.000000	1.000000	Red neuronal 0
22	class_46	1.000000	1.000000	1.000000	1.000000	Red neuronal 0
23	class_47	1.000000	0.500000	0.666667	2.000000	Red neuronal 0
24	class_5	1.000000	1.000000	1.000000	1.000000	Red neuronal 0
25	class_7	0.500000	1.000000	0.666667	1.000000	Red neuronal 0
26	class_8	1.000000	1.000000	1.000000	1.000000	Red neuronal 0
27	micro avg	0.966667	0.906250	0.935484	32.000000	Red neuronal 0
28	macro avg	0.981481	0.944444	0.950617	32.000000	Red neuronal 0
29	weighted avg	0.984375	0.906250	0.927083	32.000000	Red neuronal 0
30	1	1.000000	0.900000	0.947368	10.000000	ResNet50
31	10	1.000000	0.900000	0.947368	10.000000	ResNet50
32	11	0.909091	1.000000	0.952381	10.000000	ResNet50
33	12	0.800000	0.800000	0.800000	10.000000	ResNet50
34	13	1.000000	1.000000	1.000000	9.000000	ResNet50
35	14	0.900000	0.900000	0.900000	10.000000	ResNet50
36	15	0.875000	0.777778	0.823529	9.000000	ResNet50
37	16	0.833333	1.000000	0.909091	10.000000	ResNet50
38	17	0.909091	1.000000	0.952381	10.000000	ResNet50
39	18	0.888889	0.888889	0.888889	9.000000	ResNet50
40	19	0.750000	0.600000	0.666667	10.000000	ResNet50
41	2	1.000000	1.000000	1.000000	9.000000	ResNet50
42	21	0.900000	0.900000	0.900000	10.000000	ResNet50
43	22	0.545455	0.600000	0.571429	10.000000	ResNet50
44	23	1.000000	0.400000	0.571429	10.000000	ResNet50
45	24	1.000000	1.000000	1.000000	9.000000	ResNet50

	Métrica	precision	recall	f1-score	support	Modelo
46	25	0.750000	1.000000	0.857143	9.000000	ResNet50
47	26	1.000000	0.777778	0.875000	9.000000	ResNet50
48	27	1.000000	1.000000	1.000000	10.000000	ResNet50
49	28	0.818182	1.000000	0.900000	9.000000	ResNet50
50	29	0.909091	1.000000	0.952381	10.000000	ResNet50
51	3	0.777778	0.777778	0.777778	9.000000	ResNet50
52	30	0.818182	0.900000	0.857143	10.000000	ResNet50
53	31	0.888889	0.888889	0.888889	9.000000	ResNet50
54	32	0.900000	0.900000	0.900000	10.000000	ResNet50
55	33	0.857143	0.666667	0.750000	9.000000	ResNet50
56	34	1.000000	1.000000	1.000000	10.000000	ResNet50
57	35	0.875000	0.700000	0.777778	10.000000	ResNet50
58	36	0.769231	1.000000	0.869565	10.000000	ResNet50
59	37	0.818182	0.900000	0.857143	10.000000	ResNet50
60	38	1.000000	1.000000	1.000000	10.000000	ResNet50
61	39	0.900000	0.900000	0.900000	10.000000	ResNet50
62	4	0.888889	0.888889	0.888889	9.000000	ResNet50
63	40	1.000000	1.000000	1.000000	9.000000	ResNet50
64	41	0.818182	1.000000	0.900000	9.000000	ResNet50
65	42	1.000000	0.900000	0.947368	10.000000	ResNet50
66	43	0.875000	0.777778	0.823529	9.000000	ResNet50
67	44	0.666667	0.666667	0.666667	9.000000	ResNet50
68	45	0.833333	1.000000	0.909091	10.000000	ResNet50
69	46	1.000000	0.900000	0.947368	10.000000	ResNet50
70	47	1.000000	0.900000	0.947368	10.000000	ResNet50
71	48	1.000000	1.000000	1.000000	9.000000	ResNet50
72	5	0.888889	0.800000	0.842105	10.000000	ResNet50
73	6	0.818182	0.900000	0.857143	10.000000	ResNet50
74	7	0.777778	0.700000	0.736842	10.000000	ResNet50
75	8	0.700000	0.777778	0.736842	9.000000	ResNet50
76	9	0.818182	0.900000	0.857143	10.000000	ResNet50
77	accuracy	0.876106	0.876106	0.876106	0.876106	ResNet50

	Métrica	precision	recall	f1-score	support	Modelo
78	macro avg	0.882503	0.876359	0.873483	452.000000	ResNet50
79	weighted avg	0.882681	0.876106	0.873363	452.000000	ResNet50
80	1	1.000000	0.900000	0.947368	10.000000	MobileNetV2
81	10	0.833333	1.000000	0.909091	10.000000	MobileNetV2
82	11	0.909091	1.000000	0.952381	10.000000	MobileNetV2
83	12	0.666667	0.600000	0.631579	10.000000	MobileNetV2
84	13	0.562500	1.000000	0.720000	9.000000	MobileNetV2
85	14	0.666667	0.600000	0.631579	10.000000	MobileNetV2
86	15	0.636364	0.777778	0.700000	9.000000	MobileNetV2
87	16	0.588235	1.000000	0.740741	10.000000	MobileNetV2
88	17	0.909091	1.000000	0.952381	10.000000	MobileNetV2
89	18	0.888889	0.888889	0.888889	9.000000	MobileNetV2
90	19	0.700000	0.700000	0.700000	10.000000	MobileNetV2
91	2	0.900000	1.000000	0.947368	9.000000	MobileNetV2
92	21	0.800000	0.800000	0.800000	10.000000	MobileNetV2
93	22	0.800000	0.400000	0.533333	10.000000	MobileNetV2
94	23	0.545455	0.600000	0.571429	10.000000	MobileNetV2
95	24	0.888889	0.888889	0.888889	9.000000	MobileNetV2
96	25	0.700000	0.777778	0.736842	9.000000	MobileNetV2
97	26	0.777778	0.777778	0.777778	9.000000	MobileNetV2
98	27	0.800000	0.800000	0.800000	10.000000	MobileNetV2
99	28	0.750000	0.666667	0.705882	9.000000	MobileNetV2
100	29	0.818182	0.900000	0.857143	10.000000	MobileNetV2
101	3	0.750000	0.666667	0.705882	9.000000	MobileNetV2
102	30	0.833333	1.000000	0.909091	10.000000	MobileNetV2
103	31	0.800000	0.888889	0.842105	9.000000	MobileNetV2
104	32	0.750000	0.600000	0.666667	10.000000	MobileNetV2
105	33	0.888889	0.888889	0.888889	9.000000	MobileNetV2
106	34	1.000000	0.800000	0.888889	10.000000	MobileNetV2
107	35	0.700000	0.700000	0.700000	10.000000	MobileNetV2
108	36	0.800000	0.400000	0.533333	10.000000	MobileNetV2

	Métrica	precision	recall	f1-score	support	Modelo
109	37	0.666667	0.600000	0.631579	10.000000	MobileNetV2
110	38	1.000000	1.000000	1.000000	10.000000	MobileNetV2
111	39	0.700000	0.700000	0.700000	10.000000	MobileNetV2
112	4	0.777778	0.777778	0.777778	9.000000	MobileNetV2
113	40	0.900000	1.000000	0.947368	9.000000	MobileNetV2
114	41	1.000000	0.777778	0.875000	9.000000	MobileNetV2
115	42	1.000000	0.800000	0.888889	10.000000	MobileNetV2
116	43	1.000000	0.666667	0.800000	9.000000	MobileNetV2
117	44	0.800000	0.444444	0.571429	9.000000	MobileNetV2
118	45	0.588235	1.000000	0.740741	10.000000	MobileNetV2
119	46	1.000000	0.800000	0.888889	10.000000	MobileNetV2
120	47	0.714286	0.500000	0.588235	10.000000	MobileNetV2
121	48	0.727273	0.888889	0.800000	9.000000	MobileNetV2
122	5	1.000000	0.600000	0.750000	10.000000	MobileNetV2
123	6	0.888889	0.800000	0.842105	10.000000	MobileNetV2
124	7	0.571429	0.800000	0.666667	10.000000	MobileNetV2
125	8	0.666667	0.666667	0.666667	9.000000	MobileNetV2
126	9	0.900000	0.900000	0.900000	10.000000	MobileNetV2
127	accuracy	0.780973	0.780973	0.780973	0.780973	MobileNetV2
128	macro avg	0.799246	0.781797	0.777934	452.000000	MobileNetV2
129	weighted avg	0.799183	0.780973	0.777407	452.000000	MobileNetV2
130	1	1.000000	1.000000	1.000000	10.000000	Xception
131	10	0.909091	1.000000	0.952381	10.000000	Xception
132	11	1.000000	1.000000	1.000000	10.000000	Xception
133	12	0.800000	0.800000	0.800000	10.000000	Xception
134	13	0.750000	1.000000	0.857143	9.000000	Xception
135	14	0.857143	0.600000	0.705882	10.000000	Xception
136	15	0.727273	0.888889	0.800000	9.000000	Xception
137	16	0.666667	1.000000	0.800000	10.000000	Xception
138	17	0.909091	1.000000	0.952381	10.000000	Xception
139	18	1.000000	0.777778	0.875000	9.000000	Xception

	Métrica	precision	recall	f1-score	support	Modelo
140	19	0.888889	0.800000	0.842105	10.000000	Xception
141	2	1.000000	1.000000	1.000000	9.000000	Xception
142	21	0.900000	0.900000	0.900000	10.000000	Xception
143	22	0.700000	0.700000	0.700000	10.000000	Xception
144	23	1.000000	0.500000	0.666667	10.000000	Xception
145	24	1.000000	1.000000	1.000000	9.000000	Xception
146	25	0.818182	1.000000	0.900000	9.000000	Xception
147	26	1.000000	0.777778	0.875000	9.000000	Xception
148	27	0.833333	1.000000	0.909091	10.000000	Xception
149	28	0.900000	1.000000	0.947368	9.000000	Xception
150	29	1.000000	1.000000	1.000000	10.000000	Xception
151	3	1.000000	0.888889	0.941176	9.000000	Xception
152	30	0.909091	1.000000	0.952381	10.000000	Xception
153	31	0.900000	1.000000	0.947368	9.000000	Xception
154	32	1.000000	0.800000	0.888889	10.000000	Xception
155	33	0.875000	0.777778	0.823529	9.000000	Xception
156	34	0.909091	1.000000	0.952381	10.000000	Xception
157	35	0.875000	0.700000	0.777778	10.000000	Xception
158	36	0.909091	1.000000	0.952381	10.000000	Xception
159	37	0.727273	0.800000	0.761905	10.000000	Xception
160	38	1.000000	1.000000	1.000000	10.000000	Xception
161	39	0.818182	0.900000	0.857143	10.000000	Xception
162	4	0.888889	0.888889	0.888889	9.000000	Xception
163	40	1.000000	1.000000	1.000000	9.000000	Xception
164	41	0.888889	0.888889	0.888889	9.000000	Xception
165	42	1.000000	1.000000	1.000000	10.000000	Xception
166	43	0.888889	0.888889	0.888889	9.000000	Xception
167	44	0.857143	0.666667	0.750000	9.000000	Xception
168	45	0.833333	1.000000	0.909091	10.000000	Xception
169	46	1.000000	0.800000	0.888889	10.000000	Xception
170	47	0.700000	0.700000	0.700000	10.000000	Xception
171	48	0.900000	1.000000	0.947368	9.000000	Xception

	Métrica	precision	recall	f1-score	support	Modelo
172	5	1.000000	0.900000	0.947368	10.000000	Xception
173	6	0.900000	0.900000	0.900000	10.000000	Xception
174	7	0.727273	0.800000	0.761905	10.000000	Xception
175	8	0.750000	0.666667	0.705882	9.000000	Xception
176	9	1.000000	0.800000	0.888889	10.000000	Xception
177	accuracy	0.882743	0.882743	0.882743	0.882743	Xception
178	macro avg	0.891847	0.883215	0.880936	452.000000	Xception
179	weighted avg	0.891646	0.882743	0.880539	452.000000	Xception
180	1	1.000000	1.000000	1.000000	10.000000	IncResNetV2
181	10	1.000000	0.900000	0.947368	10.000000	IncResNetV2
182	11	1.000000	1.000000	1.000000	10.000000	IncResNetV2
183	12	0.900000	0.900000	0.900000	10.000000	IncResNetV2
184	13	1.000000	1.000000	1.000000	9.000000	IncResNetV2
185	14	0.900000	0.900000	0.900000	10.000000	IncResNetV2
186	15	0.888889	0.888889	0.888889	9.000000	IncResNetV2
187	16	0.833333	1.000000	0.909091	10.000000	IncResNetV2
188	17	0.909091	1.000000	0.952381	10.000000	IncResNetV2
189	18	1.000000	0.888889	0.941176	9.000000	IncResNetV2
190	19	0.769231	1.000000	0.869565	10.000000	IncResNetV2
191	2	1.000000	1.000000	1.000000	9.000000	IncResNetV2
192	21	1.000000	1.000000	1.000000	10.000000	IncResNetV2
193	22	0.777778	0.700000	0.736842	10.000000	IncResNetV2
194	23	0.888889	0.800000	0.842105	10.000000	IncResNetV2
195	24	1.000000	1.000000	1.000000	9.000000	IncResNetV2
196	25	0.900000	1.000000	0.947368	9.000000	IncResNetV2
197	26	0.900000	1.000000	0.947368	9.000000	IncResNetV2
198	27	0.909091	1.000000	0.952381	10.000000	IncResNetV2
199	28	1.000000	1.000000	1.000000	9.000000	IncResNetV2
200	29	1.000000	1.000000	1.000000	10.000000	IncResNetV2
201	3	1.000000	1.000000	1.000000	9.000000	IncResNetV2
202	30	0.909091	1.000000	0.952381	10.000000	IncResNetV2

	Métrica	precision	recall	f1-score	support	Modelo
203	31	1.000000	1.000000	1.000000	9.000000	IncResNetV2
204	32	1.000000	1.000000	1.000000	10.000000	IncResNetV2
205	33	0.875000	0.777778	0.823529	9.000000	IncResNetV2
206	34	0.900000	0.900000	0.900000	10.000000	IncResNetV2
207	35	1.000000	0.800000	0.888889	10.000000	IncResNetV2
208	36	0.900000	0.900000	0.900000	10.000000	IncResNetV2
209	37	0.900000	0.900000	0.900000	10.000000	IncResNetV2
210	38	0.909091	1.000000	0.952381	10.000000	IncResNetV2
211	39	0.800000	0.800000	0.800000	10.000000	IncResNetV2
212	4	1.000000	0.888889	0.941176	9.000000	IncResNetV2
213	40	1.000000	1.000000	1.000000	9.000000	IncResNetV2
214	41	1.000000	1.000000	1.000000	9.000000	IncResNetV2
215	42	1.000000	1.000000	1.000000	10.000000	IncResNetV2
216	43	1.000000	1.000000	1.000000	9.000000	IncResNetV2
217	44	0.875000	0.777778	0.823529	9.000000	IncResNetV2
218	45	0.833333	1.000000	0.909091	10.000000	IncResNetV2
219	46	1.000000	0.900000	0.947368	10.000000	IncResNetV2
220	47	0.909091	1.000000	0.952381	10.000000	IncResNetV2
221	48	1.000000	0.888889	0.941176	9.000000	IncResNetV2
222	5	0.909091	1.000000	0.952381	10.000000	IncResNetV2
223	6	1.000000	0.900000	0.947368	10.000000	IncResNetV2
224	7	0.800000	0.800000	0.800000	10.000000	IncResNetV2
225	8	0.777778	0.777778	0.777778	9.000000	IncResNetV2
226	9	1.000000	0.700000	0.823529	10.000000	IncResNetV2
227	accuracy	0.929204	0.929204	0.929204	0.929204	IncResNetV2
228	macro avg	0.933485	0.929551	0.929096	452.000000	IncResNetV2
229	weighted avg	0.932569	0.929204	0.928414	452.000000	IncResNetV2

Ahora solo obtengo las clases con un valor menor a 0.8 de precisión, recall o F1-score en todos los modelos de la estrategia 2.

```
In [ ]: # Filtrar clases con precision, recall o f1-score menor a 0.8
df_filtrado = df_comparacion[(df_comparacion["precision"] < 0.8) |
```

```

(df_comparacion["recall"] < 0.8) |
(df_comparacion["f1-score"] < 0.8))

# Contar cuántos modelos incluyen cada clase de la estrategia 2
conteo_clases = df_filtrado[df_filtrado["Modelo"] != model_names[0]].groupby

# Filtrar aquellas clases que aparecen en todos los modelos de la estrategia
clases_comunes = conteo_clases[conteo_clases == (len(model_list) - 1)].index

# Filtrar el DataFrame original para incluir solo esas clases
df_final = df_filtrado[df_filtrado["Métrica"].isin(clases_comunes)]

# Mostrar el resultado
display(df_final.style.set_caption("Clases con Métricas < 0.8 en los modelos
    .set_properties(**{'border': '1px solid black'})
    .background_gradient(subset=['precision', 'recall', 'f

```

Clases con Métricas < 0.8 en los modelos de la estrategia 2

	Métrica	precision	recall	f1-score	support	Modelo
43	22	0.545455	0.600000	0.571429	10.000000	ResNet50
67	44	0.666667	0.666667	0.666667	9.000000	ResNet50
75	8	0.700000	0.777778	0.736842	9.000000	ResNet50
93	22	0.800000	0.400000	0.533333	10.000000	MobileNetV2
117	44	0.800000	0.444444	0.571429	9.000000	MobileNetV2
125	8	0.666667	0.666667	0.666667	9.000000	MobileNetV2
143	22	0.700000	0.700000	0.700000	10.000000	Xception
167	44	0.857143	0.666667	0.750000	9.000000	Xception
175	8	0.750000	0.666667	0.705882	9.000000	Xception
193	22	0.777778	0.700000	0.736842	10.000000	IncResNetV2
217	44	0.875000	0.777778	0.823529	9.000000	IncResNetV2
225	8	0.777778	0.777778	0.777778	9.000000	IncResNetV2

```

In [ ]: #Precisión de la red neuronal de 0
val_accuracy = history.history['val_accuracy']
a=val_accuracy[-1]
x_values=model_names
y_values=[a,xception_acc,resnet_acc,mobilenet_acc,inceptionresnet_acc]

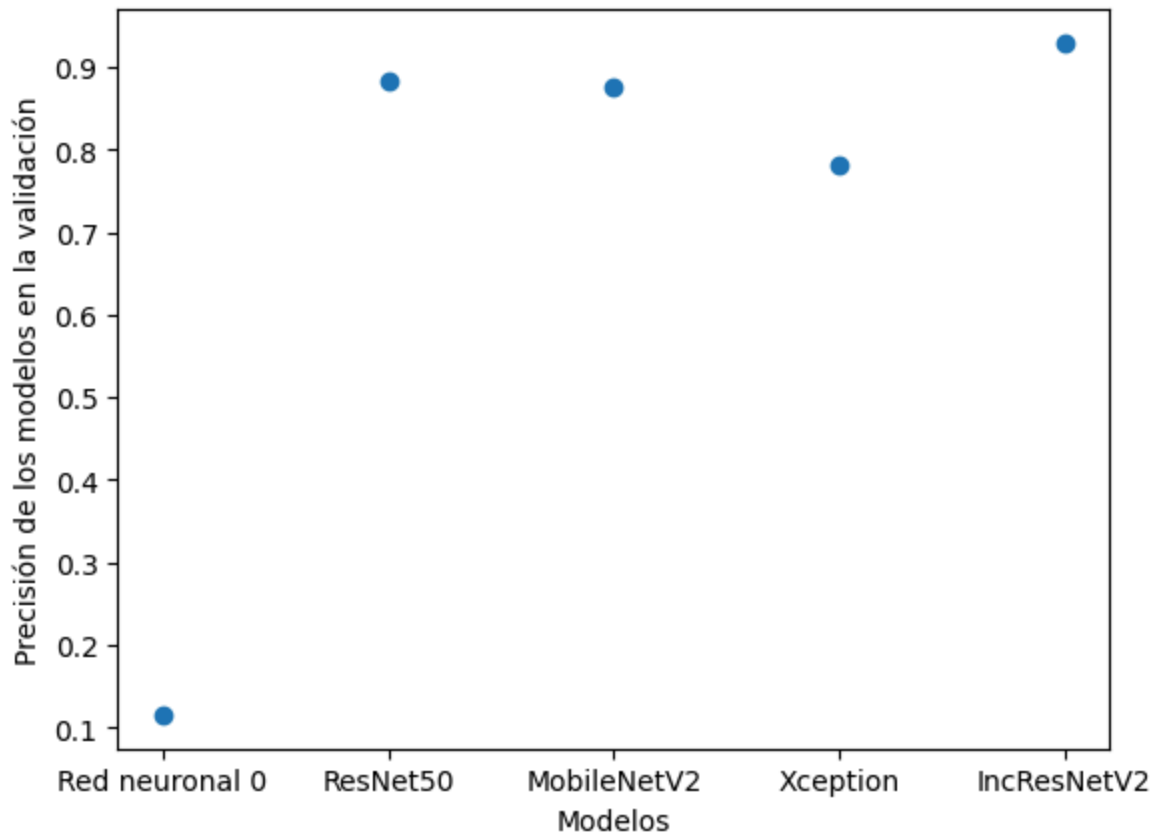
plt.scatter(x_values, y_values)
plt.xlabel("Modelos")
plt.ylabel("Precisión de los modelos en la validación")

```

```

Out[ ]: Text(0, 0.5, 'Precisión de los modelos en la validación')

```



Comparación visual de las predicciones

Durante la fase de validación, se analizan los errores cometidos por el modelo al predecir clases incorrectas. Aunque el rendimiento general sea positivo, algunos patrones de error se pueden repetir, especialmente en posturas visualmente similares entre sí o en aquellas con menor representación en el conjunto de entrenamiento.

A continuación se coge una muestra de las imágenes para ver cómo las predice cada modelo y la probabilidad con la que se calcula que forman parte de esa clase.

Solo se hará esta comparativa en la estrategia 2, ya que la de la red neuronal de 0 ha dado un resultado muy bajo de precisión y se puede descartar.

```
In [ ]: import os
import random
from PIL import Image

# Ruta con las imágenes organizadas por clase
ruta_base = "/content/DL/final"

def cargar_imagen_de_clase(y_pred):
    nombre_clase = f"class_{int(y_pred)}" # Obtener nombre de la clase
```



```

ruta_clase = os.path.join(ruta_base, nombre_clase) # Ruta de la clase

# Obtener lista de imágenes en la carpeta
imagenes_disponibles = os.listdir(ruta_clase)

if imagenes_disponibles:
    imagen_aleatoria = random.choice(imagenes_disponibles) # Elegir una
    ruta_imagen = os.path.join(ruta_clase, imagen_aleatoria)

    # Cargar imagen
    return Image.open(ruta_imagen)
else:
    print(f"No se encontraron imágenes para la clase {nombre_clase}")
    return None

```

```

In [ ]: import tensorflow as tf
import matplotlib.pyplot as plt

def resize_image(image, target_size):
    return tf.image.resize(image, target_size)

num_images = 10 # Número de imágenes a mostrar
fig, axes = plt.subplots(num_images, len(model_list) , figsize=(15, 10)) #

for img_idx in range(num_images):
    # Mostrar la imagen original con la etiqueta real
    axes[img_idx, 0].imshow(val_images[img_idx])
    axes[img_idx, 0].set_title(f"Real: {class_labels[int(y_true[img_idx])]}")
    axes[img_idx, 0].axis("off")

    # Obtener predicciones para cada modelo de la estrategia 2
    for model_idx, model in enumerate(model_list[1:]):
        input_shape = model.input_shape[1:3] # Extraer tamaño esperado
        img_resized = resize_image(val_images[img_idx], input_shape) # Redi

        # Obtener predicción del modelo
        predicciones = model.predict(np.expand_dims(img_resized, axis=0))
        y_pred = np.argmax(predicciones, axis=1)[0]
        probabilidad = predicciones[0][y_pred] # Probabilidad de la clase p



















































        # Mostrar imagen de la clase predicha si se clasificó mal
        if y_pred != y_true[img_idx]:
            imagen_predicha = cargar_imagen_de_clase(y_pred) # Función para
            axes[img_idx, model_idx + 1].imshow(imagen_predicha)
        else:
            axes[img_idx, model_idx + 1].imshow(val_images[img_idx])

        axes[img_idx, model_idx + 1].set_title(f"{model_names[model_idx + 1]}")
        axes[img_idx, model_idx + 1].axis("off")

plt.tight_layout()
plt.show()

```


[illegible]

Real: 15 	ResNet50: 14 - Prob 0.50 	MobileNetV2: 13 - Prob 0.22 	Xception: 15 - Prob 0.26 	IncResNetV2: 15 - Prob 0.88 
Real: 21 	ResNet50: 21 - Prob 0.97 	MobileNetV2: 21 - Prob 0.94 	Xception: 21 - Prob 1.00 	IncResNetV2: 21 - Prob 1.00 
Real: 47 	ResNet50: 47 - Prob 0.86 	MobileNetV2: 47 - Prob 0.15 	Xception: 47 - Prob 0.98 	IncResNetV2: 47 - Prob 0.98 
Real: 32 	ResNet50: 32 - Prob 0.85 	MobileNetV2: 32 - Prob 0.71 	Xception: 32 - Prob 1.00 	IncResNetV2: 32 - Prob 0.99 
Real: 14 	ResNet50: 15 - Prob 0.60 	MobileNetV2: 15 - Prob 0.65 	Xception: 15 - Prob 0.67 	IncResNetV2: 14 - Prob 0.92 
Real: 41 	ResNet50: 16 - Prob 0.65 	MobileNetV2: 41 - Prob 0.48 	Xception: 41 - Prob 0.84 	IncResNetV2: 41 - Prob 0.95 
Real: 3 	ResNet50: 3 - Prob 0.86 	MobileNetV2: 3 - Prob 0.91 	Xception: 3 - Prob 0.98 	IncResNetV2: 3 - Prob 1.00 
Real: 13 	ResNet50: 13 - Prob 0.99 	MobileNetV2: 13 - Prob 1.00 	Xception: 13 - Prob 1.00 	IncResNetV2: 13 - Prob 1.00 
Real: 14 	ResNet50: 15 - Prob 0.48 	MobileNetV2: 14 - Prob 0.69 	Xception: 14 - Prob 0.86 	IncResNetV2: 14 - Prob 0.99 
Real: 2 	ResNet50: 2 - Prob 0.86 	MobileNetV2: 3 - Prob 0.54 	Xception: 2 - Prob 1.00 	IncResNetV2: 2 - Prob 1.00 

Discusión y Conclusiones

Discusión

En el estudio de la **precisión de la validación** de los modelos se observa cómo el modelo InceptionResNetV2 ha sido el que ha obtenido una mayor precisión con un **0.93**. Le sigue Xception con 0.88 y ResNet50 con 0.86. Los modelos con peor precisión han sido el de la red neuronal creada de 0, con un 0.12 y el MobileNetV2 con 0.77.

Con el número de épocas que se aplicó en cada caso se pudo alcanzar un estado estacionario, en el que prácticamente no había cambios entre una época y otra. Más o menos todos alcanzaban este punto en la precisión y en la pérdida sobre las 16 épocas.

En el caso de la función de pérdida, **el modelo con una mayor pérdida es el de la red neuronal de 0**, con el doble de pérdida de validación que de entrenamiento. Esto implica que es un modelo que **no generaliza bien y tiene un sobreajuste**. Esto también se aprecia en

la mayor precisión del entrenamiento que en la validación, 0.42 y 0.12, respectivamente.

El resto de modelos tienen una pérdida de validación menor que la de entrenamiento, por lo que no hay sobreajuste. Los modelos ResNet50 e InceptionResNetV2 son los que menor pérdida tienen con 1.44 y 1.89, respectivamente.

En el **análisis de la matriz de confusión** se aprecia como en la Red neuronal de 0 prácticamente no hay distinción de clases, ni entre ellas mismas, ya que la diagonal tiene valores muy bajos de predicciones correctas. Sin embargo, en los 4 modelos de la estrategia 2 sí que hay una mayor predicción de las clases sobre ellas mismas.

En la matriz de confusión de **MobileNetV2 y ResNet50 hay muchos valores fuera de la diagonal**, lo que implica que el modelo está confundiendo entre esas clases y es probable que necesite mejoras. De la estrategia 2, estos dos fueron los que menor precisión obtuvieron, principalmente el MobileNetV2.

En cuanto a los resultados del informe de clasificación, las clases que dan una métrica (precisión, recall o F1-score) **menor de 0.8 en todos los modelos de la estrategia 2 son la 8, la 22 y la 44**; siendo la 22 la que mayor dificultades de clasificación presenta. Esto significa que estas 3 son clases difíciles y habría que analizar si requiere una mayor representación en el dataset para mejorar el entrenamiento del modelo.

La última **comparativa** del funcionamiento de los modelos ha sido **visual**. Se utilizaron funciones de predicción sobre imágenes del conjunto para comprobar visualmente si el modelo reconocía correctamente la postura. Cada imagen tiene la etiqueta predicha y la probabilidad de predicción para facilitar la interpretación. La primera columna es la imagen que los modelos tienen que clasificar.

En total se han escogido **diez** fotos reales. Solo se ha visualizado los modelos de la estrategia 2, ya que el de la estrategia 1 se puede descartar por haber dado muy malos resultados. De esta manera se ve más clara la comparativa visual.

De los **4 modelos de la estrategia 2** el único que **ha acertado todas las figuras** que tenía que clasificar es el **InceptionResNetV2**. Esto coincide con el hecho de que sea el **modelo con una mayor precisión**.

Las probabilidades con las que lo acertó fueron cercanas a 1, a excepción de la clase 15 que fue de 0.88.

El siguiente que le sigue es el **Xception**, con solo 1 fallo de clasificación en la clase 14. Sin embargo, tiene valores de probabilidad en algunos casos como la clase 15 muy bajos. **La clase 14 la clasificó como clase 15, por lo que parece tener problemas con estas 2 clases de posturas.**

El que menos clases de posturas acertó fue el modelo **ResNet50** con 4 fallos, seguido por el **MobileNetV2** con 3 fallos. **Ambos se equivocaron en la clase 15 y la clase 14.** El **InceptionResNetV2** tuvo la menor probabilidad de acierto en esta clase, por lo que **en el futuro se hace necesario aumentar el número de imágenes de la clase 15 y la clase 14.**

Los malos resultados del **MobileNet** y **ResNet50** coinciden con que son los modelos con los más bajos resultados de precisión obtenidos. Además, también son los que más valores fuera de la diagonal tienen en la matriz de confusión.

Estas visualizaciones permiten confirmar que los modelos no solo aprenden las clases durante el entrenamiento, sino que también **son capaces de predecir con éxito posturas reales del conjunto de validación**, reforzando así la utilidad práctica del sistema.

Entre los errores más comunes se observan:

- Confusión entre posturas de la clase 14 y 15, debido a la similitud en la posición corporal y orientación de los brazos.
- Dificultades en posturas inclinadas o parcialmente ocultas, donde la información visual no es clara o está afectada por ángulos poco definidos.

Estos errores reflejan tanto la complejidad del problema como la sensibilidad de los modelos a la calidad y variedad de los datos. Aun así, estos mostraron una alta tasa de aciertos en posturas bien representadas y con características visuales claras, lo que valida su utilidad para tareas prácticas de clasificación.

Conclusiones

En este trabajo se ha llevado a cabo el estudio de 5 modelos de redes neuronales con dos estrategias diferentes: hacer una red neuronal desde 0 y hacer una red neuronal basada en otras redes neuronales que han sido previamente entrenadas con un gran banco de imágenes.

El modelo de red neuronal desde 0 se creó con 3 capas convolucionales 2D regularizadas con BatchNormalization y con capas de reducción max_pooling intermedias. Después de esto se encuentran 2 capas Flatten y 2 dense con regularización Dropout. La última capa dense es la que proporciona la salida, con 47 clases.

Este modelo ha dado un resultado muy bajo, con un 0.4 de precisión con imágenes de entrenamiento y un 0.12 de precisión con imágenes de validación. Por lo tanto, no se considera un buen modelo para realizar esta tarea de clasificación de imágenes. De cara a futuros trabajos habría que estudiar diferentes configuraciones de capas para obtener mejores resultados.

En cuanto a la **estrategia 2**, se ha realizado **4 modelos** diferentes con los siguientes como **base pre-entrenada**: **ResNet50**, **Xception**, **MobileNetV2** e **InceptionResNetV2**.

ResNet50 y Xception son ideales para tareas que requieren redes profundas y complejas, mientras que MobileNetV2 es más adecuado para aplicaciones en dispositivos con recursos limitados.

InceptionResNetV2, al combinar lo mejor de Inception y ResNet, es útil para tareas de alta precisión, aunque su tamaño y complejidad lo hacen menos accesible en dispositivos con pocos recursos.

En los resultados se puede ver cómo InceptionResNetV2 ha sido el modelo con una mayor precisión y mayor número de aciertos en la comparación visual. El problema es que este modelo consume muchos recursos y requiere un tiempo excesivamente elevado para su entrenamiento.

Por los resultados de precisión, tiempo de ejecución y comparación visual, se pueden descartar los modelos de ResNet50 y MobileNetV2. Dependiendo de la necesidad de cómputo se podría elegir entre Xception y el InceptionResNetV2 para futuros trabajos, ya que son los 2 mejores modelos de los estudiados.

Exportar a PDF

```
In [ ]: !apt update && apt install -y pandoc texlive-xetex texlive-fonts-recommended
```

```
In [ ]: !apt update && apt install -y texlive-xetex texlive-fonts-recommended
```

```
In [5]: from google.colab import drive  
drive.mount('/content/drive')
```

Mounted at /content/drive

```
In [ ]: !jupyter nbconvert --to html "/content/drive/MyDrive/Colab Notebooks/Redes n
```

[NbConvertApp] Converting notebook /content/drive/MyDrive/Colab Notebooks/Redes neuronales/07MIAR_Proyecto_Programacion.ipynb to html

[NbConvertApp] WARNING | Alternative text is missing on 9 image(s).

[NbConvertApp] Writing 6135532 bytes to /content/drive/MyDrive/Colab Notebooks/Redes neuronales/07MIAR_Proyecto_Programacion.html

```
In [ ]: !pip install nbconvert[webpdf] playwright
```

```
In [ ]: !playwright install
```

Downloading Chromium 136.0.7103.25 (playwright build v1169) from https://cdn.playwright.dev/dbazure/download/playwright/builds/chromium/1169/chromium-linux.zip

167.7 MiB [] 0% 0.0s167.7 MiB [] 0% 33.9s167.7 MiB [] 0% 23.9s167.7 MiB [] 0% 15.9s167.7 MiB [] 0% 12.1s167.7 MiB [] 1% 8.6s167.7 MiB [] 1% 8.1s167.7 MiB [] 2% 6.0s167.7 MiB [] 2% 5.6s167.7 MiB [] 3% 5.1s167.7 MiB [] 3% 5.3s167.7 MiB [] 3% 5.7s167.7 MiB [] 3% 5.8s167.7 MiB [] 4% 5.7s167.7 MiB [] 4% 5.5s167.7 MiB [] 4% 5.6s167.7 MiB [] 5% 5.3s167.7 MiB [] 5% 5.5s167.7 MiB [] 6% 5.4s167.7 MiB [] 7% 5.1s167.7 MiB [] 7% 4.8s167.7 MiB [] 8% 4.8s167.7 MiB [] 8% 4.7s167.7 MiB [] 9% 4.5s167.7 MiB [] 9% 4.6s167.7 MiB [] 10% 4.5s167.7 MiB [] 11% 4.3s167.7 MiB [] 12% 4.1s167.7 MiB [] 13% 3.9s167.7 MiB [] 14% 3.6s167.7 MiB [] 14% 3.5s167.7 MiB [] 15% 3.4s167.7 MiB [] 16% 3.3s167.7 MiB [] 18% 3.1s167.7 MiB [] 18% 3.0s167.7 MiB [] 19% 3.0s167.7 MiB [] 20% 2.8s167.7 MiB [] 21% 2.8s167.7 MiB [] 21% 2.7s167.7 MiB [] 22% 2.8s167.7 MiB [] 22% 2.9s167.7 MiB [] 23% 2.8s167.7 MiB [] 24% 2.7s167.7 MiB [] 25% 2.6s167.7 MiB [] 26% 2.5s167.7 MiB [] 27% 2.4s167.7 MiB [] 28% 2.4s167.7 MiB [] 29% 2.3s167.7 MiB [] 30% 2.3s167.7 MiB [] 31% 2.2s167.7 MiB [] 32% 2.1s167.7 MiB [] 33% 2.1s167.7 MiB [] 34% 2.0s167.7 MiB [] 35% 1.9s167.7 MiB [] 36% 1.9s167.7 MiB [] 37% 1.8s167.7 MiB [] 38% 1.8s167.7 MiB [] 39% 1.7s167.7 MiB [] 40% 1.7s167.7 MiB [] 41% 1.6s167.7 MiB [] 42% 1.6s167.7 MiB [] 43% 1.5s167.7 MiB [] 45% 1.5s167.7 MiB [] 46% 1.4s167.7 MiB [] 47% 1.4s167.7 MiB [] 48% 1.3s167.7 MiB [] 49% 1.3s167.7 MiB [] 51% 1.3s167.7 MiB [] 52% 1.2s167.7 MiB [] 53% 1.2s167.7 MiB [] 54% 1.2s167.7 MiB [] 56% 1.1s167.7 MiB [] 57% 1.1s167.7 MiB [] 59% 1.0s167.7 MiB [] 60% 1.0s167.7 MiB [] 61% 0.9s167.7 MiB [] 62% 0.9s167.7 MiB [] 63% 0.9s167.7 MiB [] 64% 0.9s167.7 MiB [] 65% 0.9s167.7 MiB [] 66% 0.8s167.7 MiB [] 67% 0.8s167.7 MiB [] 68% 0.8s167.7 MiB [] 69% 0.8s167.7 MiB [] 70% 0.8s167.7 MiB [] 71% 0.7s167.7 MiB [] 71% 0.8s167.7 MiB [] 72% 0.7s167.7 MiB [] 73% 0.7s167.7 MiB [] 74% 0.7s167.7 MiB [] 75% 0.7s167.7 MiB [] 76% 0.6s167.7 MiB [] 77% 0.6s167.7 MiB [] 78% 0.6s167.7 MiB [] 79% 0.6s167.7 MiB [] 80% 0.5s167.7 MiB [] 81% 0.5s167.7 MiB [] 82% 0.5s167.7 MiB [] 83% 0.5s167.7 MiB [] 84% 0.4s167.7 MiB [] 85% 0.4s167.7 MiB [] 86% 0.4s167.7 MiB [] 87% 0.3s167.7 MiB [] 88% 0.3s167.7 MiB [] 89% 0.3s167.7 MiB [] 90% 0.3s167.7 MiB [] 91% 0.2s167.7 MiB [] 92% 0.2s167.7 MiB [] 93% 0.2s167.7 MiB [] 94% 0.2s167.7 MiB [] 94% 0.1s167.7 MiB [] 95% 0.1s167.7 MiB [] 96% 0.1s167.7 MiB [] 97% 0.1s167.7 MiB [] 98% 0.0s167.7 MiB [] 99% 0.0s167.7 MiB [] 100% 0.0s

In [6]: !jupyter nbconvert --to webpdf "/content/drive/MyDrive/Colab Notebooks/Redes

```

[NbConvertApp] Converting notebook /content/drive/MyDrive/Colab Notebooks/Red
es neuronales/07MIAR_Proyecto_Programacion.ipynb to webpdf
[NbConvertApp] WARNING | Alternative text is missing on 9 image(s).
[NbConvertApp] Building PDF
Traceback (most recent call last):
  File "/usr/local/lib/python3.11/dist-packages/nbconvert/exporters/webpdf.p
y", line 78, in main
    from playwright.async_api import async_playwright # type: ignore[import-
not-found]
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
ModuleNotFoundError: No module named 'playwright'

```

The above exception was the direct cause of the following exception:

```

Traceback (most recent call last):
  File "/usr/local/bin/jupyter-nbconvert", line 10, in <module>
    sys.exit(main())
    ^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/jupyter_core/application.py",
line 283, in launch_instance
    super().launch_instance(argv=argv, **kwargs)
  File "/usr/local/lib/python3.11/dist-packages/traitlets/config/application.
py", line 992, in launch_instance
    app.start()
  File "/usr/local/lib/python3.11/dist-packages/nbconvert/nbconvertapp.py", 1
ine 420, in start
    self.convert_notebooks()
  File "/usr/local/lib/python3.11/dist-packages/nbconvert/nbconvertapp.py", 1
ine 597, in convert_notebooks
    self.convert_single_notebook(notebook_filename)
  File "/usr/local/lib/python3.11/dist-packages/nbconvert/nbconvertapp.py", 1
ine 563, in convert_single_notebook
    output, resources = self.export_single_notebook(
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/nbconvert/nbconvertapp.py", 1
ine 487, in export_single_notebook
    output, resources = self.exporter.from_filename(
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/nbconvert/exporters/templatee
xporter.py", line 390, in from_filename
    return super().from_filename(filename, resources, **kw) # type:ignore[re
turn-value]
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/nbconvert/exporters/exporter.
py", line 201, in from_filename
    return self.from_file(f, resources=resources, **kw)
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/nbconvert/exporters/templatee
xporter.py", line 396, in from_file
    return super().from_file(file_stream, resources, **kw) # type:ignore[ret
urn-value]
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib/python3.11/dist-packages/nbconvert/exporters/exporter.
py", line 220, in from_file
    return self.from_notebook_node(
    ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

```



```
File "/usr/local/lib/python3.11/dist-packages/nbconvert/exporters/webpdf.py", line 174, in from_notebook_node
    pdf_data = self.run_playwright(html)
                ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
File "/usr/local/lib/python3.11/dist-packages/nbconvert/exporters/webpdf.py", line 163, in run_playwright
    pdf_data = pool.submit(run_coroutine, main(temp_file)).result()
                ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
File "/usr/lib/python3.11/concurrent/futures/_base.py", line 456, in result
    return self.__get_result()
           ^^^^^^^^^^^^^^^^^^
File "/usr/lib/python3.11/concurrent/futures/_base.py", line 401, in __get_result
    raise self._exception
File "/usr/lib/python3.11/concurrent/futures/thread.py", line 58, in run
    result = self.fn(*self.args, **self.kwargs)
             ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
File "/usr/local/lib/python3.11/dist-packages/nbconvert/exporters/webpdf.py", line 161, in run_coroutine
    return loop.run_until_complete(coro)
           ^^^^^^^^^^^^^^^^^^^^^^^^^^^
File "/usr/lib/python3.11/asyncio/base_events.py", line 654, in run_until_complete
    return future.result()
           ^^^^^^^^^^^^^^
File "/usr/local/lib/python3.11/dist-packages/nbconvert/exporters/webpdf.py", line 84, in main
    raise RuntimeError(msg) from e
RuntimeError: Playwright is not installed to support Web PDF conversion. Please install `nbconvert[webpdf]` to enable.
```