

ico. Este archivo se ha [Mostrar diferencias](#)

```
Requirement already satisfied: sniffio>=1.1 in /usr/local/lib/python3.11/dist-packages (from aiohttp<4.0.0,>=3.8.0->radio) (3.10)
Requirement already satisfied: certifi in /usr/local/lib/python3.11/dist-packages (from aiohttp<4.0.0,>=3.8.0->radio) (1.3.1)
Requirement already satisfied: httpcore>=1.* in /usr/local/lib/python3.11/dist-packages (from aiohttp<4.0.0,>=3.8.0->radio) (2025.4.26)
Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.11/dist-packages (from aiohttp<4.0.0,>=3.8.0->radio) (1.0.9)
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->radio) (3.1)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->radio) (2.2)
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->radio) (4.42.1)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas<3.0,>=1.0->radio) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas<3.0,>=1.0->radio) (2025.4.26)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas<3.0,>=1.0->radio) (2022.7)
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.11/dist-packages (from pydantic<2.12,>=2.0->radio) (0.6.0)
Requirement already satisfied: pydantic-core>=2.33.2 in /usr/local/lib/python3.11/dist-packages (from pydantic<2.12,>=2.0->radio) (2.33.2)
Requirement already satisfied: typing-inspection>=0.4.0 in /usr/local/lib/python3.11/dist-packages (from pydantic<2.12,>=2.0->radio) (0.4.0)
Requirement already satisfied: click>=8.0.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0,>=0.12->radio) (8.2.1)
Requirement already satisfied: shellingham>=1.3.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0,>=0.12->radio) (1.3.0)
Requirement already satisfied: rich>=10.11.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0,>=0.12->radio) (10.11.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas<3.0,>=1.0->radio) (1.5)
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.11/dist-packages (from rich>=10.11.0->typer<1.0->radio) (2.2.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from rich>=10.11.0->typer<1.0->radio) (2.13.0)
Requirement already satisfied: charset-normalizer<4,>2 in /usr/local/lib/python3.11/dist-packages (from requests->huggingface-hub>=0.1->radio) (3.2.0)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests->huggingface-hub>=0.1->radio) (1.21.1)
Requirement already satisfied: idna>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests->huggingface-hub>=0.1->radio) (3.2.0)
Requirement already satisfied: certifi>=2023.7.22 in /usr/local/lib/python3.11/dist-packages (from requests->huggingface-hub>=0.1->radio) (3.1.1)
Requirement already satisfied: chardet>=3.0.2 in /usr/local/lib/python3.11/dist-packages (from requests->huggingface-hub>=0.1->radio) (3.2.0)
Requirement already satisfied: idna>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests->huggingface-hub>=0.1->radio) (3.2.0)
Requirement already satisfied: certifi>=2023.7.22 in /usr/local/lib/python3.11/dist-packages (from requests->huggingface-hub>=0.1->radio) (3.1.1)
Requirement already satisfied: chardet>=3.0.2 in /usr/local/lib/python3.11/dist-packages (from requests->huggingface-hub>=0.1->radio) (3.2.0)
```

ico. Este archivo se ha [Mostrar diferencias](#)

2.- Cargamiento y exploración de los datos

```
[3] # Ruta en Drive
ruta_drive_train = "/content/drive/MyDrive/Deep learning/Datasets/simpsons_dataset"
ruta_drive_test = "/content/drive/MyDrive/Deep learning/Datasets/kaggle_simpson_testset"

# Nueva ruta en disco local
ruta_train = "/content/simpsons_dataset"
ruta_test = "/content/kaggle_simpson_testset"

# Copiar datasets al almacenamiento local del entorno de Colab
shutil.copytree(ruta_drive_train, ruta_train, dirs_exist_ok=True)
shutil.copytree(ruta_drive_test, ruta_test, dirs_exist_ok=True)
```

▼ Entrenamiento

```
[4] # Conteo de las carpetas del dataset de entrenamiento
os.listdir(ruta_train)

[+] ['moe_szyslak',
     'otto_mann',
     'disco_stu',
     'principal_skinner',
     'kent_brockman',
     'professor_john_frank',
     'sideshow_bob',
     'patty_bouvier',
     'charles_montgomery_burns',
     'nelson_muntz',
     'waylon_smithers',
```

ico. Este archivo se ha [Mostrar diferencias](#)

```
     'maggie_simpson',
     'fat_tony',
     'carl_carlson',
     'agnes_skinner',
     'martin_prince',
     'milhouse_van_houten',
     'mayor_quimby',
     'edna_krabappel',
     'bart_simpson',
     'cletus_spuckler',
     'ned_flanders',
     'chief_wiggum',
     'apu_nahasapeemapetilon',
     'comic_book_guy',
     'snake_jailbird',
     'marge_simpson',
     'barney_gumble',
     'krusty_the_clown',
```

ico. Este archivo se ha [Mostrar diferencias](#)

```
     'troy_mcclure',
     'lisa_simpson',
     'lionel_hutz',
     'selma_bouvier',
     'abraham_grampa_simpson']
```

```
[5] # Obtener nombres de las carpetas (clases)
```

```

        clases = os.listdir(ruta_train)
        clases = [clase for clase in clases if os.path.isdir(os.path.join(ruta_train, clase))]

    [6] # Contar imágenes por clase
    conteo_clases = {}
    for clase in clases:
        ruta_clase = os.path.join(ruta_train, clase)
        conteo_clases[clase] = len(os.listdir(ruta_clase))

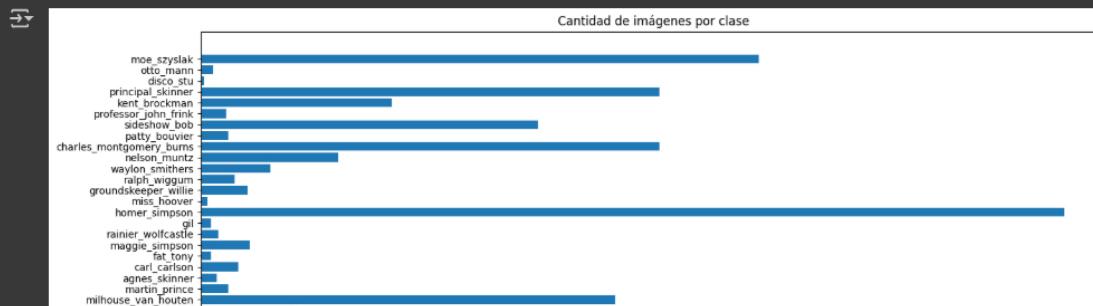
    [7] # Mostrar conteo
    for i, (clase, cantidad) in enumerate(sorted(conteo_clases.items(), key=lambda x: x[1], reverse=True), start=1):
        print(f'{i}. {clase}: {cantidad} imágenes')

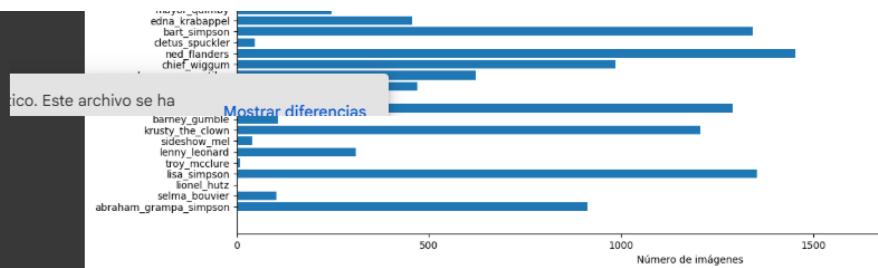
    [8] # Mostrar conteo
    for i, (clase, cantidad) in enumerate(sorted(conteo_clases.items(), key=lambda x: x[1], reverse=True), start=1):
        print(f'{i}. {clase}: {cantidad} imágenes')

    [9] # Mostrar conteo
    print(f'Total de clases (personajes): {len(conteo_clases)}')
    print(f'Total de imágenes en entrenamiento: {sum(conteo_clases.values())}')

    [10] # Visualizar en gráfico de barras
    plt.figure(figsize=(15, 8))
    plt.barh(list(conteo_clases.keys()), list(conteo_clases.values()))
    plt.xlabel('Número de imágenes')
    plt.title('Cantidad de imágenes por clase')
    plt.gca().invert_yaxis()

```





```
[11] def mostrar_ejemplos(ruta_base, clases, n=5):
    seleccionadas = random.sample(clases, n)
    plt.figure(figsize=(15, 5))
    for i, clase in enumerate(seleccionadas):
        ruta_clase = os.path.join(ruta_base, clase)
        imagen = random.choice(os.listdir(ruta_clase))
        ruta_img = os.path.join(ruta_clase, imagen)
        img = Image.open(ruta_img)
        plt.subplot(1, n, i+1)
        plt.imshow(img)
    plt.show()
```

Este archivo se ha [Mostrar diferencias](#)

mostrar_ejemplos(ruta_train, clases)



Este archivo se ha [Mostrar diferencias](#)

```
[12] # Obtener todos los archivos .jpg (ignorando archivos ocultos)
archivos = sorted([
    f for f in os.listdir(ruta_test)
    if f.lower().endswith('.jpg') and not f.startswith('.')
])

# Verificar si los nombres siguen el patrón y extraer etiquetas
etiquetas = []
archivos_validos = []

for nombre in archivos:
    match = re.match(r'^(.+)\_d+\_.jpg$', nombre)
    if match:
        etiquetas.append(match.group(1)) # nombre del personaje
        archivos_validos.append(nombre)
    else:
        print(f"⚠ Archivo ignorado por nombre no válido: {nombre}")

Este archivo se ha Mostrar diferencias
```

conteo = Counter(etiquetas)

Mostrar estadísticas generales
print(f"\n✓ Total de imágenes válidas: {len(archivos_validos)}")
print(f"✓ Total de personajes únicos: {len(conteo)}")

Mostrar todos los personajes con su número de imágenes (orden alfabético)
print("\n▣ Cantidad de imágenes por personaje:")
for personaje in sorted(conteo.keys()):
 print(f"{personaje}: {conteo[personaje]}")

▣ Cantidad de imágenes por personaje:
abraham_grampa_simpson: 48
apu_nahasapeemapetilon: 50
bart_simpson: 50
charles_montgomery_burns: 48

Este archivo se ha [Mostrar diferencias](#)

edna_krabappel: 50
homer_simpson: 50
kent_brockman: 50
krusty_the_clown: 50
lenny_leonard: 50

```

lisa_simpson: 50
marge_simpson: 50
mayor_quimby: 50
milhouse_van_houten: 49
moe_szyslak: 50
ned_flanders: 49
nelson_muntz: 50
principal_skinner: 50
sideshow_bob: 47

[13] #Mostrar 1 imagen aleatoria por personaje
muestras_unicas = {}
for archivo in archivos:
    etiqueta = re.match(r'^(.+)\d+\.\jpg$', archivo)
    if etiqueta:
        nombre_imagen = etiqueta.group(1)
        print(f'El archivo {nombre_imagen} es único.')
        muestra_unicas[nombre_imagen] = archivo

# Mostrar hasta 10 personajes con una imagen cada uno
muestras_a_mostrar = dict(random.sample(list(muestras_unicas.items()), 10))

plt.figure(figsize=(15, 6))
for i, (personaje, archivo) in enumerate(muestras_a_mostrar.items()):
    img = Image.open(os.path.join(ruta_test, archivo))
    plt.subplot(2, 5, i + 1)
    plt.imshow(img)
    plt.title(personaje, fontsize=8)
    plt.axis('off')
plt.tight_layout()
plt.show()

```



```

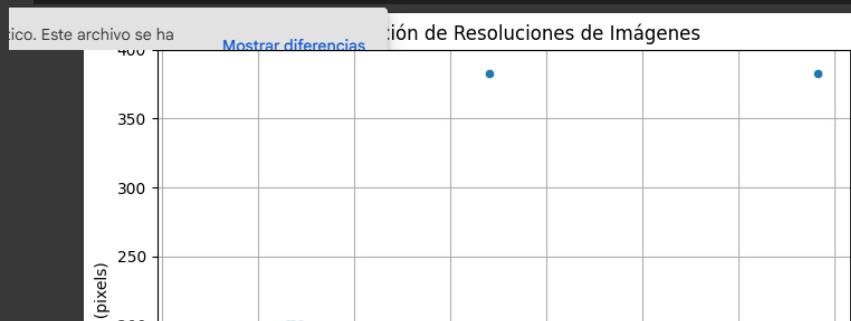
[14] # Medir ancho y alto de cada imagen
dimensiones = []

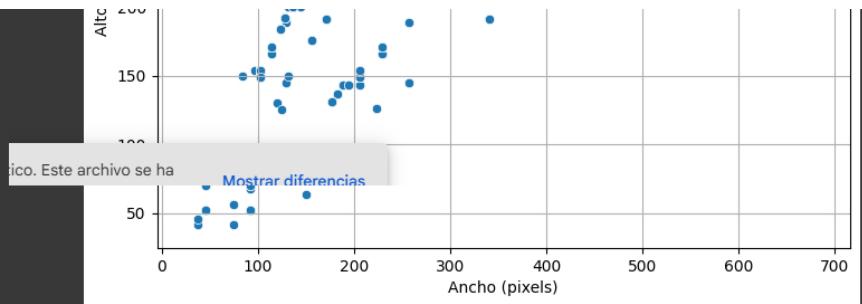
for archivo in archivos:
    ruta_img = os.path.join(ruta_test, archivo)
    print(f'Este archivo se ha leido: {ruta_img}')
    try:
        with Image.open(ruta_img) as img:
            dimensiones.append(img.size) # (ancho, alto)
    except Exception as e:
        print(f'Error al leer {archivo}: {e}')

# Convertir a numpy array
dimensiones = np.array(dimensiones)
anchos = dimensiones[:, 0]
altos = dimensiones[:, 1]

# Gráfica de dispersión ancho vs alto
plt.figure(figsize=(8, 6))
sns.scatterplot(x=anchos, y=altos)
plt.title("Distribución de Resoluciones de Imágenes")
plt.xlabel("Ancho (pixels)")
plt.ylabel("Alto (pixels)")
plt.grid(True)
plt.show()

```





3.- Procesamiento de los datos

```
[42] # 📁 Parámetros globales
IMG_SIZE = (128, 128)
BATCH_SIZE = 32
#CANTIDAD_DESEADA = 1000 # Máximo de imágenes por clase (post-aumentación)
TOP_N = 18 # Seleccionar las 18 clases más representadas

[43] # 🖼 Función para obtener las top-N clases con más imágenes
def obtener_top_clases(ruta, top_n=18):
    print("Top 18 personajes con más imágenes:")
    for clase in os.listdir(ruta):
        ruta_clase = os.path.join(ruta, clase)
        if os.path.isdir(ruta_clase):
            n_imgs = len([
                f for f in os.listdir(ruta_clase)
                if f.lower().endswith('.jpg') and not f.startswith('.')
            ])
            conteo_clases[clase] = n_imgs
    top_clases = sorted(conteo_clases, key=lambda c: conteo_clases[c], reverse=True)[:top_n]
    return top_clases, conteo_clases

# Obtener top 18 clases
top_18_clases, conteo_clases = obtener_top_clases(ruta_train, top_n=TOP_N)

# Mostrar resultado
print("Top 18 personajes con más imágenes:")
for i, clase in enumerate(top_18_clases, start=1):
    print(f"{i}. {clase}: {conteo_clases[clase]} imágenes")
```

Top 18 personajes con más imágenes:

Rank	Personaje	Cantidad de imágenes
1.	moe_szyslak	1452 imágenes
2.	lisa_simpson	1354 imágenes
3.	bart_simpson	1342 imágenes
4.	marge_simpson	1291 imágenes
5.	krusty_the_clown	1206 imágenes
6.	principal_skinner	1194 imágenes
7.	charles_montgomery_burns	1193 imágenes
8.	milhouse_van_houten	1079 imágenes
9.	chief_wiggum	986 imágenes
10.	abraham_grampa_simpson	913 imágenes
11.	sideshow_bob	877 imágenes
12.	apu_nahasapeemapetilon	623 imágenes
13.	kent_brockman	498 imágenes
14.	comic_book_guy	469 imágenes
15.	edna_krabappel	457 imágenes
16.	nelson_muntz	358 imágenes

```
[44] # 🖍️ Preparar test set
archivos = sorted([
    f for f in os.listdir(ruta_test)
    if f.lower().endswith('.jpg') and not f.startswith('.')
])

etiquetas = []
archivos_validos = []

for nombre in archivos:
    match = re.match(r'^(.+)\d+\.\jpg$', nombre)
    if match:
        etiquetas.append(match.group(1))
        archivos_validos.append(nombre)

df_test = pd.DataFrame({
    'filename': archivos_validos,
    'class': etiquetas
})

# Filtrar solo las top 18 clases
df_test = df_test[df_test['class'].isin(top_18_clases)].reset_index(drop=True)
```

```
[45] # Codificar para test
df_test['class'] = df_test['class'].map(lambda x: top_18_clases.index(x))
```

```
test_generator = test_datagen.flow_from_dataframe(
    dataframe=df_test,
    directory=ruta_test,
    x_col='filename',
    y_col='class',
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    shuffle=False,
    classes=top_18_clases
)
```

→ Found 890 validated image filenames belonging to 18 classes.

```
✓ [46] # Obtener rutas y etiquetas de imágenes de las top 18 clases directamente desde ruta_train
```

```
data_paths = []
class_labels = []

# Mostrar diferencias
ruta_clase = os.path.join(ruta_train, clase)
archivos = glob.glob(os.path.join(ruta_clase, '*.jpg'))
data_paths.extend(archivos)
class_labels.extend([clase] * len(archivos))

# Crear el DataFrame
df_all = pd.DataFrame({
    'filename': data_paths,
    'class': class_labels
})

# Separar en entrenamiento y validación (estratificado)
df_train, df_val = train_test_split(
    df_all,
    test_size=0.2,
    stratify=df_all['class'],
    random_state=42
)

# Mostrar conteo por clase
print("\nCantidad de imágenes por clase (entrenamiento):")
print(df_train['class'].value_counts())

# Mostrar conteo por clase
print("\nCantidad de imágenes por clase (validación):")
print(df_val['class'].value_counts())
```

→ Cantidad de imágenes por clase (entrenamiento):

class	count
homer_simpson	1797
ned_flanders	1163
moe_szyslak	1162
lisa_simpson	1083
bart_simpson	1074
marge_simpson	1033
krusty_the_clown	965
principal_skinner	955
charles_montgomery_burns	954
milhouse_van_houten	863
chief_wiggum	789
abraham_grampa_simpson	730
sideshow_bob	702
apu_nahasapeemapetilon	408

→ Cantidad de imágenes por clase (validación):

class	count
edna_krabappel	366
nelson_muntz	286

Name: count, dtype: int64

→ Cantidad de imágenes por clase (validación):

class	count
homer_simpson	449
ned_flanders	291
moe_szyslak	290
lisa_simpson	271
bart_simpson	268
marge_simpson	258
krusty_the_clown	241
principal_skinner	239
charles_montgomery_burns	239
milhouse_van_houten	216
chief_wiggum	197
abraham_grampa_simpson	183
sideshow_bob	175
apu_nahasapeemapetilon	125
kent_brockman	100

→ Cantidad de imágenes por clase (validación):

class	count
nelson_muntz	72

Name: count, dtype: int64

```
✓ [47] # Crear generadores
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=25,
    zoom_range=0.3,
    width_shift_range=0.2,
```

```

        height_shift_range=0.2,
        brightness_range=[0.5, 1.5],
        shear_range=0.2,
        horizontal_flip=True,
        fill_mode='nearest'
    )

# ▲ Ajustar estadísticas del generador si se usa normalización featurewise
X_sample = np.array([
    img_to_array(load_img(p, target_size=IMG_SIZE))
    for p in df_train['filename'].head(100)] # limitar por memoria

```

ico. Este archivo se ha [Mostrar diferencias](#)

```

train_datagen.fit(X_sample)

val_datagen = ImageDataGenerator(rescale=1./255) # 🔍 sin augmentación para validación

```

```

[48] train_generator = train_datagen.flow_from_dataframe(
    dataframe=df_train,
    x_col='filename',
    y_col='class',
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    shuffle=True,
    classes=top_18_clases
)

val_generator = val_datagen.flow_from_dataframe(
    dataframe=df_val,
    x_col='filename',
    y_col='class'

```

ico. Este archivo se ha [Mostrar diferencias](#)

```

    target_size=IMG_SIZE,
    class_mode='categorical',
    shuffle=False,
    classes=top_18_clases
)

```

→ Found 15193 validated image filenames belonging to 18 classes.
Found 3799 validated image filenames belonging to 18 classes.

```

[49] # 📈 Verificar balance final
conteo_balanceado = Counter(df_train['class'])

print("\n📊 Cantidad de imágenes por clase (entrenamiento - sin balancear artificialmente):")
for clase in top_18_clases:
    print(f"{clase}: {conteo_balanceado[clase]} imágenes")

```

→ 📊 Cantidad de imágenes por clase (entrenamiento - sin balancear artificialmente):
homer_simpson: 1797 imágenes
ned_flanders: 1163 imágenes

ico. Este archivo se ha [Mostrar diferencias](#)

```

bart_simpson: 1074 imágenes
marge_simpson: 1033 imágenes
krusty_the_clown: 965 imágenes
principal_skinner: 955 imágenes
charles_montgomery_burns: 954 imágenes
milhouse_van_houten: 863 imágenes
chief_wiggum: 789 imágenes
abraham_grampa_simpson: 730 imágenes
sideshow_bob: 702 imágenes
apu_nahasapeemapetilon: 498 imágenes
kent_brockman: 398 imágenes
comic_book_guy: 375 imágenes
edna_krabappel: 366 imágenes
nelson_muntz: 286 imágenes

```

```

[50] sample = df_train.sample(9)
fig, axes = plt.subplots(3, 3, figsize=(9, 9))

for img_path, label, ax in zip(sample['filename'], sample['class'], axes.flatten()):
    img = mpimg.imread(img_path)

```

ico. Este archivo se ha [Mostrar diferencias](#)

```

    ax.axis('off')

plt.tight_layout()
plt.show()

```





4.- Modelamiento

```
[84] def residual_block(x, filters, stride=1, l2_lambda=1e-4):
    shortcut = x

    # Primera convolución
    x = layers.Conv2D(filters, kernel_size=3, strides=stride, padding='same',
                      kernel_regularizer=regularizers.l2(l2_lambda))(x)
    x = layers.BatchNormalization()(x)
    x = layers.LeakyReLU()(x)

    # Segunda convolución
    x = layers.Conv2D(filters, kernel_size=3, strides=1, padding='same',
                      kernel_regularizer=regularizers.l2(l2_lambda))(x)
    x = layers.BatchNormalization()(x)

    # Shortcut si cambia el tamaño o número de filtros
    if stride != 1 or shortcut.shape[-1] != filters:
        shortcut = layers.Conv2D(filters, kernel_size=1, strides=stride, padding='same',
                               kernel_regularizer=regularizers.l2(l2_lambda))(shortcut)
        shortcut = layers.BatchNormalization()(shortcut)

    # Suma residual + activación + dropout
    x = layers.add([x, shortcut])
    x = layers.LeakyReLU()(x)
    x = layers.Dropout(0.3)(x)

    return x

[85] def build_resnet(input_shape=(128, 128, 3), num_classes=18, l2_lambda=1e-4):
    inputs = Input(shape=input_shape, name="Input")

    # Capa inicial
    x = layers.Conv2D(64, kernel_size=7, strides=2, padding='same',
                      kernel_regularizer=regularizers.l2(l2_lambda), name="ConvInicial")(inputs)
    x = layers.BatchNormalization()(x)
    x = layers.LeakyReLU()(x)
    x = layers.MaxPooling2D(pool_size=3, strides=2, padding='same')(x)

    # Bloques residuales
    x = residual_block(x, filters=64, stride=1, l2_lambda=l2_lambda)
    x = residual_block(x, filters=64, stride=1, l2_lambda=l2_lambda)

    x = residual_block(x, filters=128, stride=2, l2_lambda=l2_lambda)
    x = residual_block(x, filters=128, stride=1, l2_lambda=l2_lambda)

    x = residual_block(x, filters=256, stride=2, l2_lambda=l2_lambda)
    x = residual_block(x, filters=256, stride=1, l2_lambda=l2_lambda)

    final
    x = layers.GlobalAvgPool2D()(x)
    x = layers.Dense(512, kernel_regularizer=regularizers.l2(l2_lambda))(x)
    x = layers.BatchNormalization()(x)
    x = layers.LeakyReLU()(x)
    x = layers.Dropout(0.5)(x)

    outputs = layers.Dense(num_classes, activation='softmax',
```

```

        kernel_regularizer=regularizers.l2(l2_lambda),
        name="Output")(x)

    model = models.Model(inputs, outputs, name="ResNet_Mejorada_v3")
    return model

```

0s [86] # Crear modelo
model = build_resnet(input_shape=(128, 128, 3), num_classes=18)

Ver arquitectura
model.summary()

	conv2d_89 (Conv2D)	(None, 8, 8, 256)	33,024	dropout_40[0][0]
batch_normalizatio...	(BatchNormalizatio...	(None, 8, 8, 256)	1,024	conv2d_87[0][0]
add_37 (Add)	(Add)	(None, 8, 8, 256)	0	batch_normalizat...
leaky_re_lu_84 (LeakyReLU)	(LeakyReLU)	(None, 8, 8, 256)	0	add_37[0][0]
dropout_41 (Dropout)	(Dropout)	(None, 8, 8, 256)	0	leaky_re_lu_84[0...
conv2d_89 (Conv2D)	(Conv2D)	(None, 8, 8, 256)	590,080	dropout_41[0][0]
batch_normalizatio...	(BatchNormalizatio...	(None, 8, 8, 256)	1,024	conv2d_89[0][0]
leaky_re_lu_85 (LeakyReLU)	(LeakyReLU)	(None, 8, 8, 256)	0	batch_normalizat...
batch_normalizatio...	(BatchNormalizatio...	(None, 8, 8, 256)	590,080	leaky_re_lu_85[0...
add_38 (Add)	(Add)	(None, 8, 8, 256)	0	conv2d_90[0][0]
leaky_re_lu_86 (LeakyReLU)	(LeakyReLU)	(None, 8, 8, 256)	0	batch_normalizat...
dropout_42 (Dropout)	(Dropout)	(None, 8, 8, 256)	0	dropout_41[0][0]
global_average_poo...	(GlobalAveragePool...	(None, 256)	0	leaky_re_lu_86[0...
dense_7 (Dense)	(Dense)	(None, 512)	131,584	global_average_p...
batch_normalizatio...	(BatchNormalizatio...	(None, 512)	2,048	dense_7[0][0]
leaky_re_lu_87 (LeakyReLU)	(LeakyReLU)	(None, 512)	0	batch_normalizat...
dropout_43 (Dropout)	(Dropout)	(None, 18)	0	leaky_re_lu_87[0...
Output (Dense)	(Dense)	(None, 18)	9,234	dropout_43[0][0]

Total params: 2,932,370 (11.19 MB)
Trainable params: 2,926,866 (11.17 MB)
Non-trainable params: 5,504 (21.50 KB)

5.- Entrenamiento

0s [87] # Definimos el optimizador SGD con Nesterov
optimizador = SGD(learning_rate=0.01, momentum=0.9, nesterov=True)

Compilamos el modelo con pérdida y métrica de precisión
model.compile(
optimizer=optimizador,
loss='categorical_crossentropy',

0s [88] class EpochSaver(tf.keras.callbacks.Callback):
"""
Guarda en un archivo la última época completada, útil para reanudar entrenamientos.
"""
def __init__(self, path):
super().__init__()
self.path = path

def on_epoch_end(self, epoch, logs=None):
with open(self.path, 'w') as f:
f.write(str(epoch + 1)) # Guarda la próxima época

0s [89] # Definir rutas base para guardar checkpoints y logs

```
checkpoint_dir = "/content/drive/MyDrive/Deep learning/Checkpoints"
log_base_dir = "/content/drive/MyDrive/Deep learning/Logs"

# Rutas fijas
# Mostrar diferencias
# ico. Este archivo se ha
# Mostrar diferencias
log_dir = os.path.join(log_base_dir, "resnet_logs") # Logs de TensorBoard

# Crear callback personalizado para guardar la época
epoch_saver = EpochSaver(epoch_file_path)

[90] # Detiene el entrenamiento si el modelo deja de mejorar
early_stopping = EarlyStopping(
    monitor='val_loss',
    patience=5,
    restore_best_weights=True,
    verbose=1
)

# Guarda el mejor modelo basado en la precisión de validación
model_checkpoint = ModelCheckpoint(
    filepath=checkpoint_path,
    monitor='val_accuracy',
    save_best_only=True,
    verbose=1
)
# Mostrar diferencias
# ico. Este archivo se ha
# Mostrar diferencias

# Reduce la tasa de aprendizaje si la pérdida no mejora
reduce_lr = ReduceLROnPlateau(
    monitor='val_loss',
    factor=0.5,
    patience=2,
    min_lr=1e-6,
    verbose=1
)
# Mostrar diferencias
# ico. Este archivo se ha
# Mostrar diferencias

# Guarda métricas para visualización en TensorBoard
tensorboard = TensorBoard(
    log_dir=log_dir,
    histogram_freq=1
)
# Mostrar diferencias
# ico. Este archivo se ha
# Mostrar diferencias

# Lista completa de callbacks
callbacks = [
    early_stopping,
    model_checkpoint,
    epoch_saver
]
# Mostrar diferencias
# ico. Este archivo se ha
# Mostrar diferencias

[91] # Obtener las clases del generador
class_indices = train_generator.class_indices
classes = list(class_indices.keys()) # nombres de clases

# Obtener etiquetas verdaderas desde el generador
y_train = train_generator.classes # array de etiquetas numéricas (0, 1, ..., 17)

# Calcular los pesos balanceados
pesos = compute_class_weight(
    class_weight='balanced',
    classes=np.unique(y_train),
    y=y_train
)

# Crear el diccionario class_weight
class_weight = dict(zip(np.unique(y_train), pesos))
# Mostrar diferencias
# ico. Este archivo se ha
# Mostrar diferencias

[92] # Entrenamos el modelo con los generadores de datos
historia = model.fit(
    train_generator,
    validation_data=val_generator,
    epochs=50,
    callbacks=callbacks,
    steps_per_epoch=train_generator.samples // train_generator.batch_size,
    validation_steps=val_generator.samples // val_generator.batch_size,
    class_weight=class_weight,
    verbose=1
)
# Verificamos si se usaron todas las épocas o se detuvo por EarlyStopping
epocas_planificadas = historia.params['epochs']
epocas_ejecutadas = len(historia.history['loss'])

if epocas_ejecutadas < epocas_planificadas:
    print("⚠️ Entrenamiento detenido anticipadamente por EarlyStopping en la época [epocas_ejecutadas].")
else:
    print("✅ Entrenamiento completado hasta la última época.")

# Mostrar diferencias
# ico. Este archivo se ha
# Mostrar diferencias
```

```

474/474 7s 14ms/step - accuracy: 0.5625 - loss: 1.5722 - val_accuracy: 0.7418 - val_loss: 1.0518 - le
Epoch 29/50
474/474 0s 210ms/step - accuracy: 0.6469 - loss: 1.3044
Epoch 29: val_accuracy improved from 0.74947 to 0.76245, saving model to /content/drive/MyDrive/Deep learning/Checkpoints/
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file f
474/474 143s 245ms/step - accuracy: 0.6469 - loss: 1.3044 - val_accuracy: 0.7624 - val_loss: 1.0157 -
Epoch 30/50
1/474 21s 46ms/step - accuracy: 0.5938 - loss: 1.7777
Epoch 30: val_accuracy did not improve from 0.76245
474/474 11s 23ms/step - accuracy: 0.5938 - loss: 1.7777 - val_accuracy: 0.7595 - val_loss: 1.0158 -
Epoch 31/50
474/474 0s 212ms/step - accuracy: 0.6509 - loss: 1.2751
Epoch 31: val_accuracy improved from 0.76245 to 0.76271, saving model to /content/drive/MyDrive/Deep learning/Checkpoints/
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file f
474/474 142s 269ms/step - accuracy: 0.6509 - loss: 1.2751 - val_accuracy: 0.7627 - val_loss: 1.0122 -
Epoch 32/50
1/474 21s 45ms/step - accuracy: 0.5312 - loss: 1.2139
Epoch 32: val_accuracy did not improve from 0.76271
    s/step - accuracy: 0.5312 - loss: 1.2139 - val_accuracy: 0.7619 - val_loss: 1.0104 - le
    ico. Este archivo se ha Mostrar diferencias
474/474 0s 207ms/step - accuracy: 0.6467 - loss: 1.3097
Epoch 33: val_accuracy improved from 0.76271 to 0.77701, saving model to /content/drive/MyDrive/Deep learning/Checkpoints/
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file f
474/474 143s 285ms/step - accuracy: 0.6467 - loss: 1.3097 - val_accuracy: 0.7770 - val_loss: 0.9513 -
Epoch 34/50
1/474 21s 45ms/step - accuracy: 0.6562 - loss: 1.3424
Epoch 34: val_accuracy improved from 0.77701 to 0.77728, saving model to /content/drive/MyDrive/Deep learning/Checkpoints/
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file f
474/474 8s 16ms/step - accuracy: 0.6562 - loss: 1.3424 - val_accuracy: 0.7773 - val_loss: 0.9516 -
Epoch 35/50
474/474 0s 209ms/step - accuracy: 0.6563 - loss: 1.2830
Epoch 35: val_accuracy did not improve from 0.77728
    Epoch 35: ReduceLROnPlateau reducing learning rate to 7.812499825377017e-05.
474/474 166s 226ms/step - accuracy: 0.6563 - loss: 1.2830 - val_accuracy: 0.7142 - val_loss: 1.1600 -
Epoch 36/50
1/474 21s 46ms/step - accuracy: 0.7188 - loss: 0.9263
Epoch 36: val_accuracy did not improve from 0.77728
474/474 7s 14ms/step - accuracy: 0.7188 - loss: 0.9263 - val_accuracy: 0.7135 - val_loss: 1.1607 - le
Epoch 37/50
474/474 0s 207ms/step - accuracy: 0.6574 - loss: 1.2253
    ico. Este archivo se ha Mostrar diferencias
    Epoch 37: ReduceLROnPlateau reducing learning rate to 3.9062499126885086e-05.
474/474 142s 239ms/step - accuracy: 0.6574 - loss: 1.2253 - val_accuracy: 0.7685 - val_loss: 0.9667 -
Epoch 38/50
1/474 21s 46ms/step - accuracy: 0.6250 - loss: 1.3427
Epoch 38: val_accuracy did not improve from 0.77728
474/474 8s 16ms/step - accuracy: 0.6250 - loss: 1.3427 - val_accuracy: 0.7685 - val_loss: 0.9661 - le
Epoch 38: early stopping
Restoring model weights from the end of the best epoch: 33.
⚠️ Entrenamiento detenido anticipadamente por EarlyStopping en la época 38.

```

```

✓ [93] # --- Solo si se requiere reanudar entrenamiento ---
"""
start_epoch = 0

# Cargar la última época entrenada si existe
if os.path.exists(epoch_file_path):
    with open(epoch_file_path, 'r') as f:
        start_epoch = int(f.read())
    print(f"🕒 Reanudando desde la época {start_epoch}")

    ico. Este archivo se ha Mostrar diferencias
    if os.path.exists(checkpoint_path):
        model.load_weights(checkpoint_path)
        print("✅ Pesos del modelo cargados desde el checkpoint")

    # Reanudar entrenamiento desde la última época
    historia = model.fit(
        train_generator,
        validation_data=val_generator,
        initial_epoch=start_epoch, # ⚡ ;Clave para reanudar!
        epochs=50,
        callbacks=callbacks
    )
"""

🕒 'nstart_epoch = 0\n# Cargar la última época entrenada si existe\nif os.path.exists(epoch_file_path):\n    with open(epoch_file_path, \'r\') as f:\n        start_epoch = int(f.read())\n        print(f"🕒 Reanudando desde la época {start_epoch}\n")\n# Cargar pesos guardados si existen\nif os.path.exists(checkpoint_path):\n    model.load_weights(checkpoint_path)\nprint("✅ Pesos del modelo cargados desde el checkpoint")\n# Reanudar entrenamiento desde la última época\nhistoria = model.fit(\n    train_generator,\n    validation_data=val_generator,\n    initial_epoch=start_epoch, # ⚡ ;Clave para reanudar!\n    epochs=50,\n    callbacks=callbacks\n)\n'

```

6.- Evaluacion del modelo

```

✓ [94] # Reiniciar el generador para asegurar que comience desde el principio
test_generator.reset() # ✅ Paso 2

# Inicializar listas para almacenar etiquetas verdaderas y predichas
y_true = []
y_pred = []

# Iteramos sobre el generador de prueba para obtener predicciones

```

```

for batch_images, batch_labels in test_generator:
    # Predecimos sin mostrar barra de progreso
    batch_preds = model.predict(batch_images, verbose=0) # ✅ Paso 1

    # Convertimos one-hot a indices de clase
    y_true.extend(np.argmax(batch_labels, axis=1))
    y_pred.extend(np.argmax(batch_preds, axis=1))

# Mostrar diferencias
if len(y_true) >= test_generator.samples:
    break

```

ico. Este archivo se ha [Mostrar diferencias](#)

```

[95] # Nombres de las clases (etiquetas)
class_names = list(test_generator.class_indices.keys())

# Calcular la matriz de confusión
cm = confusion_matrix(y_true, y_pred)

# Visualización con seaborn
plt.figure(figsize=(12, 10))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
            xticklabels=class_names, yticklabels=class_names)
plt.xlabel('Predicción')
plt.ylabel('Etiqueta verdadera')
plt.title('Matriz de Confusión')
plt.xticks(rotation=45)
plt.yticks(rotation=45)
plt.tight_layout()

```

ico. Este archivo se ha [Mostrar diferencias](#)

```

>>> <ipython-input-95-cbc2e47eb36>:16: UserWarning: Glyph 128269 (\N{LEFT-POINTING MAGNIFYING GLASS}) missing from font(s) Deja
      plt.tight_layout()
/usr/local/lib/python3.11/dist-packages/IPython/core/pylabtools.py:151: UserWarning: Glyph 128269 (\N{LEFT-POINTING MAGNIFYI
fig.canvas.print_figure(bytes_io, **kw)

```

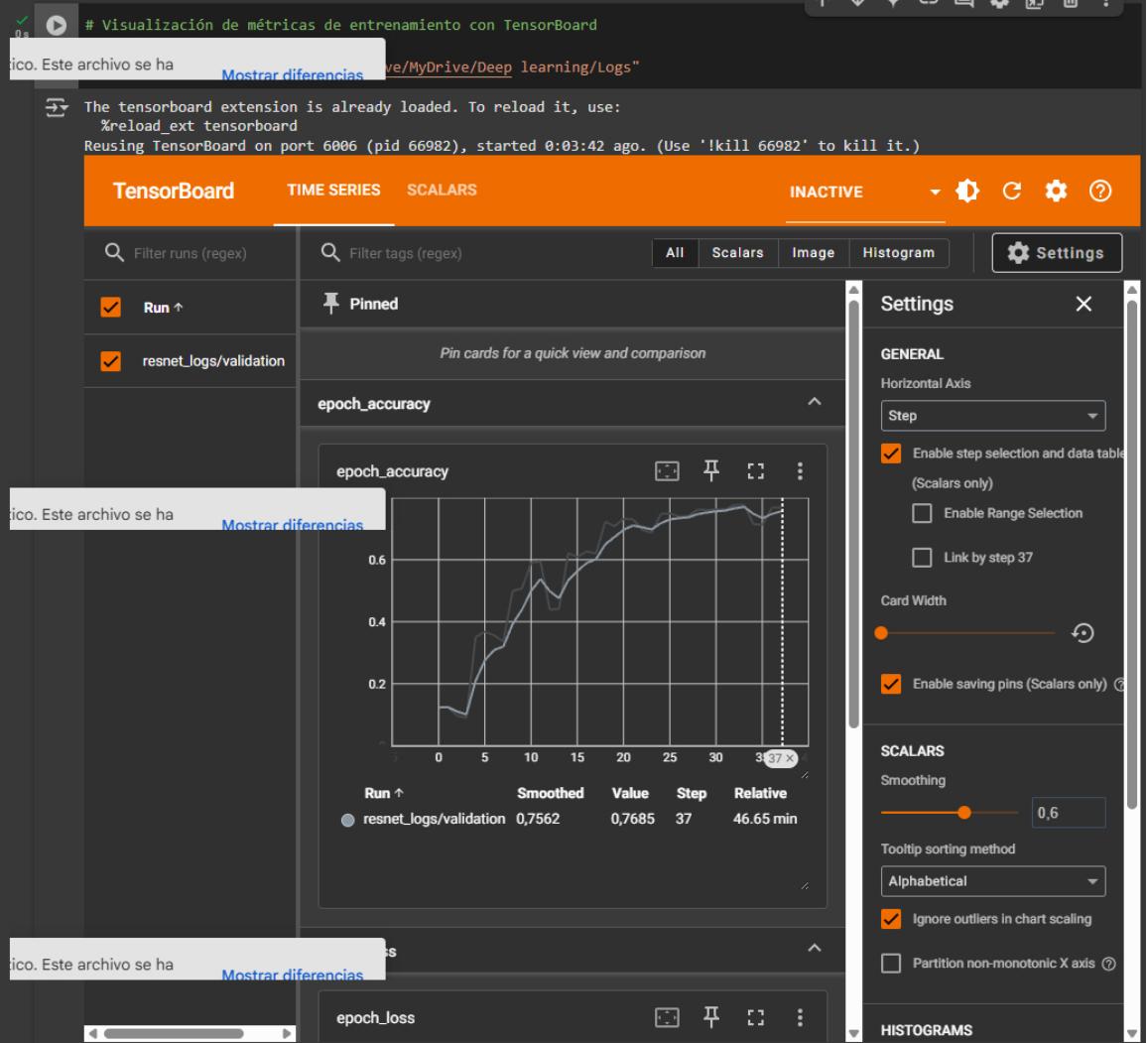
Matriz de Confusión

		Matriz de Confusión																														
		homer_simpson	ned_flanders	moe_szyslak	lisa_simpson	bart_simpson	marge_simpson	krusty_the_clown	skinner	charles_montgomery_burns	milhouse_van_houten	chief_wiggum	apu_nahasapeemah	abraham_grampa	kent_brockman	comic_book_guy	edna_krabappel	nelson_muntz	lisa_simpson	bart_simpson	marge_simpson	krusty_the_clown	principal_skinner	charles_montgomery_burns	milhouse_van_houten	chief_wiggum	apu_nahasapeemah	abraham_grampa	kent_brockman	comic_book_guy	edna_krabappel	nelson_muntz
		homer_simpson	35	1	0	1	5	1	1	1	0	0	1	0	0	0	0	0	3	0												
Etiqueta verdadera		ned_flanders	1	43	0	0	1	0	0	0	0	0	0	0	0	0	0	0	2	2												
		moe_szyslak	4	0	36	3	0	0	0	2	1	0	0	1	0	0	2	0	1	0												
		lisa_simpson	3	1	0	30	11	0	1	0	0	0	0	2	0	0	0	0	2	0												
		bart_simpson	2	0	1	2	33	0	1	1	1	0	0	1	0	0	2	0	3	3												
		marge_simpson	1	1	0	0	0	46	0	0	0	0	0	0	1	0	0	0	1	0												
		krusty_the_clown	0	0	0	0	0	0	46	0	0	0	0	0	1	0	0	0	2	0												
		skinner	-	-	-	0	3	0	0	38	2	0	0	0	1	0	0	0	0	0												
		charles_montgomery_burns	0	0	1	0	0	0	0	40	0	0	0	0	1	0	0	0	1	1												
		milhouse_van_houten	0	1	0	0	0	1	0	0	0	45	1	1	0	0	0	0	0	0												
		chief_wiggum	0	0	2	0	0	0	1	0	0	0	45	1	0	0	0	0	0	0												
		apu_nahasapeemah	5	0	2	0	0	0	0	2	0	0	0	38	1	0	0	0	0	0												
		abraham_grampa	0	0	1	0	0	0	0	0	0	0	0	45	1	0	0	0	0	0												
		kent_brockman	0	0	0	0	0	0	0	0	0	0	0	38	1	0	0	0	0	0												
		comic_book_guy	0	0	1	1	1	0	0	0	0	0	0	43	2	0	0	0	1	0												
		edna_krabappel	0	2	0	0	0	0	0	8	4	0	2	3	1	1	0	0	26	0												
		nelson_muntz	0	4	1	0	1	0	0	0	0	0	0	0	1	0	0	0	41	0												

Classification Report:

	precision	recall	f1-score	support
homer_simpson	0.60	0.70	0.65	50
ned_flanders	0.75	0.88	0.81	49
moe_szyslak	0.77	0.72	0.74	50
lisa_simpson	0.60	0.69	0.60	50
bart_simpson	0.65	0.63	0.64	50

	Mostrar diferencias	0.80	0.82	50
marge_simson	0.94	0.92	0.93	50
krusty_the_clown	0.92	0.92	0.92	50
principal_skinner	0.72	0.76	0.74	50
charles_montgomery_burns	0.80	0.83	0.82	48
milhouse_van_houten	0.98	0.92	0.95	49
chief_wiggum	0.92	0.90	0.91	50
abraham_grampa_simpson	0.72	0.79	0.75	48
sideshow_bob	0.91	0.91	0.91	47
apu_nahasapeemapetilon	0.86	0.98	0.92	50
kent_brockman	0.90	0.88	0.89	50
comic_book_guy	1.00	0.53	0.69	49
edna_krabappel	0.72	0.82	0.77	50
nelson_muntz	0.79	0.76	0.78	50
accuracy			0.80	890
macro avg	0.82	0.80	0.80	890
weighted avg	0.82	0.80	0.80	890



```
[98] # Evaluar el modelo directamente sobre el generador de test
loss, accuracy = model.evaluate(test_generator)
print(f" Precisión del modelo en el conjunto de prueba: {accuracy:.4f}")

3/28 [0s 27ms/step - accuracy: 0.8264 - loss: 0.8034/usr/local/lib/python3.11/dist-packages/keras/src/_util.py:145: UserWarning: self._warn_if_super_not_called()
28/28 [2s 73ms/step - accuracy: 0.8044 - loss: 0.8912
Precisión del modelo en el conjunto de prueba: 0.8045
```

7.- Despliegue

ticos. Este archivo se ha Mostrar diferencias "/content/MyDrive/Deep learning/Checkpoints/mejor_modelo_resnet.h5"

```
# Diccionario con los nombres de los personajes
class_indices = train_generator.class_indices # Asegúrate de que esté definido
idx_to_class = {v: k for k, v in class_indices.items()}

WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty
```

```

0s [100] del predecir_personajes_en_video(video_input):
    # Obtener ruta del video

    video_path = None
    if isinstance(video_input, str):
        video_path = video_input
    elif hasattr(video_input, 'name'): # Para archivos subidos en Gradio
        video_path = video_input.name

    if video_path is None or not os.path.exists(video_path):
        return "Error: No se pudo obtener la ruta del video o el archivo no existe."
    else:
        print(f"Este archivo se ha cargado: {video_path}")

    if not cap.isOpened():
        return "Error: No se pudo abrir el video."

    personajes_detectados = []

    frame_count = 0
    inference_img_size = (128, 128)

    # Define cuántos frames saltar. Saltar más frames hace la detección más rápida pero puede perder apariciones cortas.
    frame_skip = 5

    # Usar tqdm para mostrar progreso si es posible
    total_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))

    print(f"Procesando video: {video_path}")
    print(f"Total de frames: {total_frames}")

    # Inicializar tqdm
    with tqdm(total=total_frames, desc="Procesando frames del video") as pbar:
        while True:
            if cap.read()[0]:
                print(f"Este archivo se ha leído: {cap.read()[0]}")
                break

            # Procesar cada frame_skip frames
            if frame_count % frame_skip == 0:
                try:
                    # Redimensionar y normalizar la imagen
                    img_resized = cv2.resize(frame, inference_img_size)
                    # Convertir de BGR a RGB si el modelo espera RGB (la mayoría de modelos entrenados con Keras/TF en imágenes)
                    img_rgb = cv2.cvtColor(img_resized, cv2.COLOR_BGR2RGB)
                    img_normalized = img_rgb / 255.0
                    input_array = np.expand_dims(img_normalized, axis=0)

                    # Realizar la predicción
                    pred = model.predict(input_array, verbose=0)[0] # predict returns a batch of predictions, take the first one

                    # Obtener la clase predicha
                    clase_idx = np.argmax(pred)
                    nombre = idx_to_class.get(clase_idx, "Desconocido") # Usar .get para manejar índices no encontrados

                    if nombre:
                        print(f"Nombre: {nombre} - Confidence: {pred[clase_idx]}")

                    # Añadir un umbral de confianza si solo quieres las predicciones más seguras
                    confidence_threshold = 0.5 # Ejemplo: solo agregar si la confianza es > 50%
                    if pred[clase_idx] > confidence_threshold:
                        personajes_detectados.append(nombre)

                except Exception as e:
                    print(f"Error procesando frame {frame_count}: {e}")
                    # Continuar con el siguiente frame aunque haya un error
                    pass

                frame_count += 1
                pbar.update(1) # Actualiza la barra de progreso

            cap.release()

    if not personajes_detectados:
        return "No se detectaron personajes con confianza suficiente en el video."
    else:
        print(f"\nPersonajes detectados en el video (ordenados por frecuencia):\n\n{personajes_detectados}")
        resultado = ""
        for personaje, cantidad in conteo.most_common():
            resultado += f"✓ {personaje}: {cantidad} apariciones\n"

        return resultado

```

```

2s [102] # Interfaz Gradio
if 'model' in globals() and 'idx_to_class' in globals():
    interface = gr.Interface(
        fn=predecir_personajes_en_video,
        inputs=gr.Video(label="Sube un video"),
        outputs=gr.Textbox(label="Personajes detectados"),
        title="Detector de Personajes de Los Simpson",
        description="El modelo analizará el video y mostrará qué personajes detectó, contando las apariciones en los frames",
        allow_flagging='never' # Desactiva la opción de 'Flag' en Gradio
    )
    interface.launch(share=True)

```

```
else:  
    print("Error: El modelo o el diccionario de clases no están cargados. Asegúrate de ejecutar las secciones anteriores.")  
ico. Este archivo se ha  
warnings.warn()  
Colab notebook detected. To show errors in colab notebook, set debug=True in launch()  
* Running on public URL: https://f45ed04bb053bfb5bc.gradio.live  
This share link expires in 1 week. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the  


## Detector de Personajes de Los Simpsons



El modelo analizará el video y mostrará qué personajes detectó, contando las apariciones en los frames procesados.



Sube un video  Mostrar diferencias Upload Reset X



Clear Submit



Personajes detectados



- Personajes detectados en el video (ordenados por frecuencia):
- ned_flanders: 32 apariciones
- homer_simpson: 28 apariciones
- bart_simpson: 27 apariciones
- marge_simpson: 22 apariciones
- lisa_simpson: 17 apariciones
- moe_szyslak: 6 apariciones
- nelson_muntz: 6 apariciones



Usar vía API  · Construido con Gradio  · Configuración



actualizado de forma remota o en otra pestaña. Mostrar diferencias Productos de pago de Colab - Cancelar contratos


```