```
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
import os
import tarfile
import seaborn as sns
from sklearn.metrics import confusion_matrix
import numpy as np
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
from sklearn.utils.class_weight import compute_class_weight

# Enable mixed precision training
tf.keras.mixed_precision.set_global_policy('mixed_float16')

# Set image size and batch size
IMG_HEIGHT = 96
IMG_WIDTH = 96
BATCH_SIZE = 32

# Download and Extract the Caltech-101 Dataset
_URL = 'https://data.caltech.edu/records/mzrjq-6wc02/files/caltech-101.zip'
path_to_zip = tf.keras.utils.get_file('caltech101.zip', origin=_URL, extract=True)
extracted_path = os.path.dirname(path_to_zip)
parent_dir = os.path.join(extracted_path, 'caltech-101')

# Extract the tar.gz file
tar_path = os.path.join(parent_dir, '101_ObjectCategories.tar.gz')
if tarfile.is_tarfile(tar_path):
    with tarfile.open(tar_path, 'r:gz') as tar:
        tar.extractall(path=parent_dir)

# Set data directory to the extracted folder
data_dir = os.path.join(parent_dir, '101_ObjectCategories')

# Data Augmentation and Preprocessing
train_datagen = ImageDataGenerator(
    rescale=1.0/255,
    validation_split=0.2,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True,
    brightness_range=[0.9, 1.1],
    fill_mode='nearest'
)

validation_datagen = ImageDataGenerator(rescale=1.0/255, validation_split=0.2)

# Training and validation generators
train_generator = train_datagen.flow_from_directory(
    data_dir,
    target_size=(IMG_HEIGHT, IMG_WIDTH),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    subset='training'
)

validation_generator = validation_datagen.flow_from_directory(
    data_dir,
    target_size=(IMG_HEIGHT, IMG_WIDTH),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    subset='validation'
)

# Compute class weights
class_weights = compute_class_weight(
    'balanced',
    classes=np.unique(train_generator.classes),
    y=train_generator.classes
)
class_weight_dict = dict(enumerate(class_weights))

# Model Definition
base_model = MobileNetV2(input_shape=(IMG_HEIGHT, IMG_WIDTH, 3), include_top=False, weights='imagenet')

# Freeze all layers for initial training
base_model.trainable = False
```

```
# Build the model
model = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(512, activation='relu'),
    layers.BatchNormalization(),
    layers.Dropout(0.2),  # Reduced dropout
    layers.Dense(256, activation='relu'),
    layers.BatchNormalization(),
    layers.Dropout(0.1),  # Reduced dropout
    layers.Dense(train_generator.num_classes, activation='softmax', dtype='float32')
])

# Optimizer and loss
optimizer = tf.keras.optimizers.Adam(learning_rate=0.001)  # Faster convergence
loss = tf.keras.losses.CategoricalCrossentropy(label_smoothing=0.1)

# Compile the model
model.compile(optimizer=optimizer, loss=loss, metrics=['accuracy'])

# Early stopping and learning rate reduction
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
lr_scheduler = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=3)

# Train the model
history = model.fit(
    train_generator,
    validation_data=validation_generator,
    epochs=30,  # Increased epochs
    callbacks=[early_stopping, lr_scheduler],
    class_weight=class_weight_dict
)

# Confusion matrix
y_pred = model.predict(validation_generator)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true = validation_generator.classes

cm = confusion_matrix(y_true, y_pred_classes)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')

# Plot training and validation accuracy and loss
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs_range = range(len(acc))

plt.figure(figsize=(8, 8))
plt.subplot(1, 2, 1)
plt.plot(epochs_range, acc, label='Training Accuracy')
plt.plot(epochs_range, val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.title('Training and Validation Accuracy')

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()

# Evaluate the model
test_loss, test_acc = model.evaluate(validation_generator)
print(f'Test Accuracy: {test_acc:.2f}')
```

```
Downloading data from https://data.caltech.edu/records/mzrjq-6wc02/files/caltech-101.zip
137414764/137414764 ━━━━━━━━━━━━━━━━━━━━ 3s 0us/step
Found 7356 images belonging to 102 classes.
Found 1788 images belonging to 102 classes.
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_weights_tf_dim_ordering
9406464/9406464 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Epoch 1/30
/usr/local/lib/python3.10/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:122: UserWarning: Your `PyDataset` cl
  self._warn_if_super_not_called()
230/230 ━━━━━━━━━━━━━━━━━━━━ 362s 2s/step - accuracy: 0.3613 - loss: 3.5018 - val_accuracy: 0.7002 - val_loss: 1.9075 - learning_rat
Epoch 2/30
230/230 ━━━━━━━━━━━━━━━━━━━━ 400s 2s/step - accuracy: 0.7538 - loss: 1.7254 - val_accuracy: 0.7657 - val_loss: 1.7148 - learning_rat
Epoch 3/30
230/230 ━━━━━━━━━━━━━━━━━━━━ 360s 2s/step - accuracy: 0.7923 - loss: 1.5657 - val_accuracy: 0.7545 - val_loss: 1.7582 - learning_rat
Epoch 4/30
230/230 ━━━━━━━━━━━━━━━━━━━━ 399s 2s/step - accuracy: 0.8234 - loss: 1.4833 - val_accuracy: 0.7931 - val_loss: 1.5697 - learning_rat
Epoch 5/30
230/230 ━━━━━━━━━━━━━━━━━━━━ 368s 2s/step - accuracy: 0.8469 - loss: 1.3869 - val_accuracy: 0.7802 - val_loss: 1.6401 - learning_rat
Epoch 6/30
230/230 ━━━━━━━━━━━━━━━━━━━━ 371s 2s/step - accuracy: 0.8520 - loss: 1.3410 - val_accuracy: 0.7724 - val_loss: 1.6377 - learning_rat
Epoch 7/30
230/230 ━━━━━━━━━━━━━━━━━━━━ 394s 2s/step - accuracy: 0.8581 - loss: 1.3410 - val_accuracy: 0.7819 - val_loss: 1.6051 - learning_rat
Epoch 8/30
230/230 ━━━━━━━━━━━━━━━━━━━━ 349s 2s/step - accuracy: 0.8817 - loss: 1.2620 - val_accuracy: 0.8070 - val_loss: 1.5036 - learning_rat
Epoch 9/30
230/230 ━━━━━━━━━━━━━━━━━━━━ 401s 2s/step - accuracy: 0.9035 - loss: 1.2083 - val_accuracy: 0.7975 - val_loss: 1.5203 - learning_rat
Epoch 10/30
230/230 ━━━━━━━━━━━━━━━━━━━━ 349s 2s/step - accuracy: 0.9094 - loss: 1.2071 - val_accuracy: 0.7975 - val_loss: 1.5262 - learning_rat
Epoch 11/30
230/230 ━━━━━━━━━━━━━━━━━━━━ 350s 2s/step - accuracy: 0.9067 - loss: 1.1721 - val_accuracy: 0.8098 - val_loss: 1.4863 - learning_rat
Epoch 12/30
230/230 ━━━━━━━━━━━━━━━━━━━━ 348s 2s/step - accuracy: 0.9076 - loss: 1.1623 - val_accuracy: 0.8154 - val_loss: 1.4923 - learning_rat
Epoch 13/30
230/230 ━━━━━━━━━━━━━━━━━━━━ 349s 2s/step - accuracy: 0.9195 - loss: 1.1410 - val_accuracy: 0.7964 - val_loss: 1.5456 - learning_rat
Epoch 14/30
230/230 ━━━━━━━━━━━━━━━━━━━━ 381s 2s/step - accuracy: 0.9108 - loss: 1.1627 - val_accuracy: 0.8121 - val_loss: 1.4723 - learning_rat
Epoch 15/30
230/230 ━━━━━━━━━━━━━━━━━━━━ 402s 2s/step - accuracy: 0.9321 - loss: 1.1042 - val_accuracy: 0.8020 - val_loss: 1.5099 - learning_rat
Epoch 16/30
230/230 ━━━━━━━━━━━━━━━━━━━━ 347s 2s/step - accuracy: 0.9289 - loss: 1.1347 - val_accuracy: 0.8143 - val_loss: 1.4710 - learning_rat
```

```
import tensorflow as tf
from tensorflow.python.client import device_lib
print(device_lib.list_local_devices())
tf.config.list_physical_devices('GPU')
```

```
[]
Epoch 21/30
230/230 ━━━━━━━━━━━━━━━━━━━━ 383s 2s/step - accuracy: 0.9317 - loss: 1.1153 - val_accuracy: 0.8037 - val_loss: 1.4904 - learning_rat
Epoch 22/30
230/230 ━━━━━━━━━━━━━━━━━━━━ 381s 2s/step - accuracy: 0.9392 - loss: 1.0932 - val_accuracy: 0.8098 - val_loss: 1.5232 - learning_rat
Epoch 23/30
230/230 ━━━━━━━━━━━━━━━━━━━━ 401s 2s/step - accuracy: 0.9395 - loss: 1.0946 - val_accuracy: 0.8087 - val_loss: 1.4786 - learning_rat
Epoch 24/30
230/230 ━━━━━━━━━━━━━━━━━━━━ 348s 2s/step - accuracy: 0.9343 - loss: 1.0766 - val_accuracy: 0.8210 - val_loss: 1.4409 - learning_rat
Epoch 25/30
230/230 ━━━━━━━━━━━━━━━━━━━━ 368s 2s/step - accuracy: 0.9487 - loss: 1.0581 - val_accuracy: 0.8199 - val_loss: 1.4517 - learning_rat
Epoch 26/30
230/230 ━━━━━━━━━━━━━━━━━━━━ 348s 2s/step - accuracy: 0.9515 - loss: 1.0414 - val_accuracy: 0.8188 - val_loss: 1.4440 - learning_rat
Epoch 27/30
230/230 ━━━━━━━━━━━━━━━━━━━━ 368s 2s/step - accuracy: 0.9473 - loss: 1.0416 - val_accuracy: 0.8233 - val_loss: 1.4378 - learning_rat
Epoch 28/30
230/230 ━━━━━━━━━━━━━━━━━━━━ 348s 2s/step - accuracy: 0.9542 - loss: 1.0401 - val_accuracy: 0.8177 - val_loss: 1.4615 - learning_rat
Epoch 29/30
230/230 ━━━━━━━━━━━━━━━━━━━━ 402s 2s/step - accuracy: 0.9499 - loss: 1.0477 - val_accuracy: 0.8154 - val_loss: 1.4619 - learning_rat
Epoch 30/30
230/230 ━━━━━━━━━━━━━━━━━━━━ 373s 2s/step - accuracy: 0.9492 - loss: 1.0350 - val_accuracy: 0.8238 - val_loss: 1.4342 - learning_rat
56/56 ━━━━━━━━━━━━━━━━━━━━ 68s 1s/step
```