```
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

# 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')

# Verify the data directory
print(f"Data directory: {data_dir}")

# Check the directory structure to verify that the data is extracted correctly
print("Files in the dataset directory:")
print(os.listdir(data_dir))

# Data Augmentation and Preprocessing
train_datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

# 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_datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2,
)

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

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

# Use MobileNetV2 as a pre-trained feature extractor
base_model = MobileNetV2(input_shape=(IMG_HEIGHT, IMG_WIDTH, 3), include_top=False, weights='imagenet')

# Freeze all base model layers
base_model.trainable = False
```

```
# Build the model
model = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(512, activation='relu'),
    layers.BatchNormalization(),
    layers.Dropout(0.5),
    layers.Dense(train_generator.num_classes, activation='softmax', dtype='float32')
])

# Compile the model
optimizer = tf.keras.optimizers.Adam(learning_rate=0.002)
model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accuracy'])

# Display the model summary
model.summary()

# Early stopping and learning rate reduction
early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
lr_scheduler = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=5)

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

# Confusion matrix to show the distribution of correct and incorrect classifications
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 based on validation data
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
Data directory: /root/.keras/datasets/caltech-101/101_ObjectCategories
Files in the dataset directory:
['dolphin', 'lobster', 'laptop', 'dragonfly', 'anchor', 'butterfly', 'dollar_bill', 'cougar_body', 'stegosaurus', 'elephant', 'ac
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_order
9406464/9406464 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Model: "sequential"
```

Layer (type)	Output Shape	Param #
mobilenetv2_1.00_96 (Functional)	(None, 3, 3, 1280)	2,257,984
global_average_pooling2d (GlobalAveragePooling2D)	(None, 1280)	0
dense (Dense)	(None, 512)	655,872
batch_normalization (BatchNormalization)	(None, 512)	2,048
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 102)	52,326

```
 Total params: 2,968,230 (11.32 MB)
 Trainable params: 709,222 (2.71 MB)
 Non-trainable params: 2,259,008 (8.62 MB)
Epoch 1/30
/usr/local/lib/python3.10/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:122: UserWarning: Your `PyDataset`
  self._warn_if_super_not_called()
230/230 ━━━━━━━━━━━━━━━━━━━━ 369s 2s/step - accuracy: 0.5063 - loss: 2.4059 - val_accuracy: 0.7886 - val_loss: 0.8220 - learning_
Epoch 2/30
230/230 ━━━━━━━━━━━━━━━━━━━━ 411s 2s/step - accuracy: 0.7917 - loss: 0.7995 - val_accuracy: 0.8182 - val_loss: 0.7112 - learning_
Epoch 3/30
230/230 ━━━━━━━━━━━━━━━━━━━━ 389s 2s/step - accuracy: 0.8229 - loss: 0.6581 - val_accuracy: 0.8093 - val_loss: 0.7215 - learning_
Epoch 4/30
230/230 ━━━━━━━━━━━━━━━━━━━━ 360s 2s/step - accuracy: 0.8303 - loss: 0.6156 - val_accuracy: 0.8188 - val_loss: 0.6726 - learning_
Epoch 5/30
230/230 ━━━━━━━━━━━━━━━━━━━━ 397s 2s/step - accuracy: 0.8434 - loss: 0.5526 - val_accuracy: 0.8266 - val_loss: 0.6736 - learning_
Epoch 6/30
230/230 ━━━━━━━━━━━━━━━━━━━━ 359s 2s/step - accuracy: 0.8508 - loss: 0.5126 - val_accuracy: 0.8317 - val_loss: 0.6745 - learning_
Epoch 7/30
230/230 ━━━━━━━━━━━━━━━━━━━━ 383s 2s/step - accuracy: 0.8630 - loss: 0.4777 - val_accuracy: 0.8194 - val_loss: 0.6757 - learning_
Epoch 8/30
230/230 ━━━━━━━━━━━━━━━━━━━━ 360s 2s/step - accuracy: 0.8781 - loss: 0.4176 - val_accuracy: 0.8305 - val_loss: 0.6741 - learning_
Epoch 9/30
230/230 ━━━━━━━━━━━━━━━━━━━━ 376s 2s/step - accuracy: 0.8838 - loss: 0.4161 - val_accuracy: 0.8372 - val_loss: 0.6781 - learning_
Epoch 10/30
230/230 ━━━━━━━━━━━━━━━━━━━━ 367s 2s/step - accuracy: 0.8953 - loss: 0.3559 - val_accuracy: 0.8473 - val_loss: 0.6259 - learning_
Epoch 11/30
230/230 ━━━━━━━━━━━━━━━━━━━━ 363s 2s/step - accuracy: 0.9021 - loss: 0.3183 - val_accuracy: 0.8322 - val_loss: 0.6554 - learning_
Epoch 12/30
230/230 ━━━━━━━━━━━━━━━━━━━━ 381s 2s/step - accuracy: 0.9030 - loss: 0.3148 - val_accuracy: 0.8473 - val_loss: 0.6118 - learning_
Epoch 13/30
230/230 ━━━━━━━━━━━━━━━━━━━━ 377s 2s/step - accuracy: 0.9122 - loss: 0.2683 - val_accuracy: 0.8350 - val_loss: 0.6829 - learning_
Epoch 14/30
```