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

# Enable Mixed Precision Training
from tensorflow.keras.mixed_precision import set_global_policy
set_global_policy('mixed_float16')
print("Mixed precision policy set to 'mixed_float16'.")

# Set image size and batch size
IMG_HEIGHT = 96
IMG_WIDTH = 96
BATCH_SIZE = 16

# 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)
dataset_dir = path_to_zip.replace('caltech101.zip', '')

# Extract the dataset from the zip file
shutil.unpack_archive(path_to_zip, dataset_dir)

# Verify extraction of tar.gz files inside the extracted directory
parent_dir = os.path.join(dataset_dir, 'caltech-101')
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)

data_dir = os.path.join(parent_dir, '101_ObjectCategories')

# Data Augmentation and Preprocessing
train_datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2,
    rotation_range=40,
    width_shift_range=0.3,
    height_shift_range=0.3,
    zoom_range=0.3,
    horizontal_flip=True,
    brightness_range=[0.8, 1.2]
)

validation_datagen = ImageDataGenerator(
    rescale=1./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',
    shuffle=False
)

# Compute Class Weights for Imbalanced Classes
class_weights = compute_class_weight(
    'balanced',
    classes=np.unique(train_generator.classes),
    y=train_generator.classes
```

```
)
class_weights_dict = dict(enumerate(class_weights))

# Learning Rate Scheduler (Cyclical Learning Rate)
def clr_schedule(epoch):
    base_lr = 0.0001
    max_lr = 0.001
    step_size = 5
    cycle = np.floor(1 + epoch / (2 * step_size))
    x = np.abs(epoch / step_size - 2 * cycle + 1)
    lr = base_lr + (max_lr - base_lr) * np.maximum(0, (1 - x))
    return lr

clr = LearningRateScheduler(clr_schedule)

# Optimized CNN Model Architecture for Faster Training
model = models.Sequential([
    # Input Layer
    Input(shape=(64, 64, 3)),  # Reduced input size for faster computation

    # First Convolutional Block
    layers.Conv2D(16, (3, 3), activation='relu'),
    layers.BatchNormalization(),
    layers.MaxPooling2D((2, 2)),

    # Second Convolutional Block
    layers.Conv2D(32, (3, 3), activation='relu'),
    layers.BatchNormalization(),
    layers.MaxPooling2D((2, 2)),

    # Third Convolutional Block
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.BatchNormalization(),
    layers.MaxPooling2D((2, 2)),

    # Global Average Pooling
    layers.GlobalAveragePooling2D(),

    # Fully Connected Dense Layer
    layers.Dense(128, activation='relu'),  # Reduced size for faster training
    layers.Dropout(0.4),
    layers.Dense(train_generator.num_classes, activation='softmax', dtype='float32')  # Explicitly cast to float32
])

# Compile the optimized model
optimizer = tf.keras.optimizers.Adam(learning_rate=0.001)  # Slightly higher LR for faster convergence
model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accuracy'])

# Model summary
model.summary()


# Callbacks
early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
lr_scheduler = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=3, min_lr=1e-6)
model_checkpoint = ModelCheckpoint(
    'optimized_model.keras',
    monitor='val_loss',
    save_best_only=True,
    verbose=1
)

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

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

class_labels = list(validation_generator.class_indices.keys())
cm = confusion_matrix(y_true, y_pred_classes)

plt.figure(figsize=(10, 8))
sns.heatmap(cm, annot=True, fmt='d', xticklabels=class_labels, yticklabels=class_labels)
plt.title('Confusion Matrix')
plt.ylabel('True Labels')
```

```
plt.xlabel('Predicted Labels')
plt.show()

# 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=(12, 6))
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 on validation data
test_loss, test_acc = model.evaluate(validation_generator)
print(f'Test Accuracy: {test_acc:.4f}')
```

```
Mixed precision policy set to 'mixed_float16'.
Downloading data from https://data.caltech.edu/records/mzrjq-6wc02/files/caltech-101.zip
137414764/137414764 ━━━━━━━━━━━━━━━━━━━━ 5s 0us/step
Found 7356 images belonging to 102 classes.
Found 1788 images belonging to 102 classes.
Model: "sequential"
```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 62, 62, 16)	448
batch_normalization (BatchNormalization)	(None, 62, 62, 16)	64
max_pooling2d (MaxPooling2D)	(None, 31, 31, 16)	0
conv2d_1 (Conv2D)	(None, 29, 29, 32)	4,640
batch_normalization_1 (BatchNormalization)	(None, 29, 29, 32)	128
max_pooling2d_1 (MaxPooling2D)	(None, 14, 14, 32)	0
conv2d_2 (Conv2D)	(None, 12, 12, 64)	18,496
batch_normalization_2 (BatchNormalization)	(None, 12, 12, 64)	256
max_pooling2d_2 (MaxPooling2D)	(None, 6, 6, 64)	0
global_average_pooling2d (GlobalAveragePooling2D)	(None, 64)	0
dense (Dense)	(None, 128)	8,320
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 102)	13,158

```
 Total params: 45,510 (177.77 KB)
 Trainable params: 45,286 (176.90 KB)
 Non-trainable params: 224 (896.00 B)
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()
460/460 ━━━━━━━━━━━━━━━━━━━━ 0s 4s/step - accuracy: 0.0112 - loss: 4.7241
Epoch 1: val_loss improved from inf to 4.56286, saving model to optimized_model.keras
460/460 ━━━━━━━━━━━━━━━━━━━━ 1650s 4s/step - accuracy: 0.0112 - loss: 4.7240 - val_accuracy: 0.0201 - val_loss: 4.5629 - learning
Epoch 2/30
460/460 ━━━━━━━━━━━━━━━━━━━━ 0s 4s/step - accuracy: 0.0226 - loss: 4.5812
Epoch 2: val_loss improved from 4.56286 to 4.43971, saving model to optimized_model.keras
460/460 ━━━━━━━━━━━━━━━━━━━━ 1699s 4s/step - accuracy: 0.0226 - loss: 4.5811 - val_accuracy: 0.0794 - val_loss: 4.4397 - learning
Epoch 3/30
460/460 ━━━━━━━━━━━━━━━━━━━━ 0s 4s/step - accuracy: 0.0636 - loss: 4.3615
Epoch 3: val_loss improved from 4.43971 to 4.42389, saving model to optimized_model.keras
460/460 ━━━━━━━━━━━━━━━━━━━━ 1704s 4s/step - accuracy: 0.0636 - loss: 4.3615 - val_accuracy: 0.0917 - val_loss: 4.4239 - learning
Epoch 4/30
460/460 ━━━━━━━━━━━━━━━━━━━━ 0s 4s/step - accuracy: 0.0982 - loss: 4.2992
Epoch 4: val_loss improved from 4.42389 to 4.19257, saving model to optimized_model.keras
460/460 ━━━━━━━━━━━━━━━━━━━━ 1703s 4s/step - accuracy: 0.0982 - loss: 4.2991 - val_accuracy: 0.1018 - val_loss: 4.1926 - learning
Epoch 5/30
460/460 ━━━━━━━━━━━━━━━━━━━━ 0s 4s/step - accuracy: 0.1072 - loss: 4.1639
Epoch 5: val_loss did not improve from 4.19257
460/460 ━━━━━━━━━━━━━━━━━━━━ 1703s 4s/step - accuracy: 0.1072 - loss: 4.1639 - val_accuracy: 0.0464 - val_loss: 4.7859 - learning
Epoch 6/30
460/460 ━━━━━━━━━━━━━━━━━━━━ 0s 4s/step - accuracy: 0.1482 - loss: 4.0723
Epoch 6: val_loss did not improve from 4.19257
460/460 ━━━━━━━━━━━━━━━━━━━━ 1703s 4s/step - accuracy: 0.1482 - loss: 4.0722 - val_accuracy: 0.0688 - val_loss: 4.3641 - learning
Epoch 7/30
460/460 ━━━━━━━━━━━━━━━━━━━━ 0s 4s/step - accuracy: 0.1621 - loss: 3.9315
Epoch 7: val_loss improved from 4.19257 to 3.95222, saving model to optimized_model.keras
460/460 ━━━━━━━━━━━━━━━━━━━━ 1649s 4s/step - accuracy: 0.1621 - loss: 3.9316 - val_accuracy: 0.1415 - val_loss: 3.9522 - learning
Epoch 8/30
 64/460 ━━━━━━━━━━━━━━━━━━━━ 23:34 4s/step - accuracy: 0.1808 - loss: 3.8264
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-1-f407476e6b88> in <cell line: 145>()
    143
    144 # Train the model with 30 epochs
--> 145 history = model.fit(
    146     train_generator,
    147     epochs=30,

                                    10 frames
/usr/local/lib/python3.10/dist-packages/tensorflow/python/eager/execute.py in quick_execute(op_name, num_outputs, inputs, attrs,
ctx, name)
     51   try:
     52     ctx.ensure_initialized()
---> 53     tensors = pywrap_tfe.TFE_Py_Execute(ctx._handle, device_name, op_name,
     54                                         inputs, attrs, num_outputs)
```

```
        except core._NotOkStatusException as e:
```

```
import os

def print_directory_structure(root_dir, level=0):
    for item in os.listdir(root_dir):
        item_path = os.path.join(root_dir, item)
        if os.path.isdir(item_path):
            print('  ' * level + f'{item}/')
            print_directory_structure(item_path, level + 1)
        else:
            print('  ' * level + f'{item}')

dataset_dir = '/root/.keras/datasets/caltech-101'
print_directory_structure(dataset_dir)
```

```
import tarfile
import os

# Set up paths
dataset_dir = '/root/.keras/datasets/caltech-101'
tar_path = os.path.join(dataset_dir, '101_ObjectCategories.tar.gz')

# Extract files
with tarfile.open(tar_path, 'r:gz') as tar:
    tar.extractall(path=dataset_dir)

# Print the new directory structure
print_directory_structure(dataset_dir)
```

```
import os
import shutil

dataset_dir = '/root/.keras/datasets/caltech-101'
images_dir = os.path.join(dataset_dir, '101_ObjectCategories')

# Create subdirectories for each class if they don't exist
for image in os.listdir(images_dir):
    if image.endswith('.jpg'):
        class_name = image.split('_')[0]  # Assuming the filename format: class_image.jpg
        class_dir = os.path.join(images_dir, class_name)
        if not os.path.exists(class_dir):
            os.makedirs(class_dir)
        shutil.move(os.path.join(images_dir, image), os.path.join(class_dir, image))

print_directory_structure(images_dir)
```

```
import os

data_dir = '/root/.keras/datasets/101_ObjectCategories'
print(os.listdir(data_dir))
```

```
import os

parent_dir = '/root/.keras/datasets'
print(os.listdir(parent_dir))
```

```
train_generator = train_datagen.flow_from_directory(
    '/root/.keras/datasets/caltech-101',  # Corrected path
    target_size=(IMG_HEIGHT, IMG_WIDTH),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    subset='training'
)

validation_generator = ImageDataGenerator(rescale=1./255).flow_from_directory(
    '/root/.keras/datasets/caltech-101',
    target_size=(IMG_HEIGHT, IMG_WIDTH),
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    subset='validation'
)
```

```
import os

def print_directory_structure(root_dir, level=0):
    for item in os.listdir(root_dir):
```