```
from google.colab import drive
import zipfile
import os

# Step 1: Mount Google Drive
drive.mount('/content/drive')

# Step 2: Define the path to the ZIP file in your Google Drive
zip_file_path = '/content/drive/MyDrive/zip-folder.zip'  # Replace with your actual file path
extract_to_path = '/content/extracted_files'  # Directory where the ZIP file will be extracted

# Step 3: Create the extraction directory if it doesn't exist
os.makedirs(extract_to_path, exist_ok=True)

# Step 4: Extract the ZIP file
with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
    zip_ref.extractall(extract_to_path)

print(f"Files extracted to: {extract_to_path}")
```

```
Mounted at /content/drive
Files extracted to: /content/extracted_files
```

```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import MobileNet
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.models import load_model


# Set paths to your dataset folders
train_dir = '/content/extracted_files/train'
val_dir = '/content/extracted_files/val'
test_dir = '/content/extracted_files/test'


# Load the model
model = load_model('/content/drive/MyDrive/mobilenet_finetuned.keras')
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/saving/saving_lib.py:713: UserWarning: Skipping varia
  saveable.load_own_variables(weights_store.get(inner_path))
```

```
# Hyperparameters
batch_size = 32
image_size = (224, 224)  # MobileNet requires 224x224 input size
epochs = 20
learning_rate = 0.001


# Data Augmentation for Training
train_datagen = ImageDataGenerator(
    rescale=1.0 / 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_datagen = ImageDataGenerator(rescale=1.0 / 255)
test_datagen = ImageDataGenerator(rescale=1.0 / 255)


# Data Generators
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=image_size,
    batch_size=batch_size,
    class_mode='categorical'
)

val_generator = val_datagen.flow_from_directory(
    val_dir,
    target_size=image_size,
    batch_size=batch_size,
    class_mode='categorical'
)

test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=image_size,
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False
)
```

```
Found 6953 images belonging to 100 classes.
Found 1966 images belonging to 100 classes.
Found 1034 images belonging to 100 classes.
```

```
train_loss, train_accuracy = model.evaluate(train_generator)
print(f"Train Loss: {train_loss:.4f}\tTrain Accuracy: {train_accuracy * 100:.2f}%")

test_loss, test_accuracy = model.evaluate(test_generator)
print(f"Test Loss: {test_loss:.4f}\tTest Accuracy: {test_accuracy * 100:.2f}%")
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:122: Use
  self._warn_if_super_not_called()
218/218 ━━━━━━━━━━━━━━━━━━━━ 100s 426ms/step - accuracy: 0.9891 - loss: 0.0341
Train Loss: 0.0335      Train Accuracy: 98.94%
33/33 ━━━━━━━━━━━━━━━━━━━━ 6s 177ms/step - accuracy: 0.9983 - loss: 0.0104
Test Loss: 0.0071       Test Accuracy: 99.90%
```

```
model.compile(optimizer=Adam(learning_rate=learning_rate / 100),
              loss='categorical_crossentropy',
              metrics=['accuracy'])

history_final_tune = model.fit(
    train_generator,
    epochs=5,
    validation_data=val_generator
)
```

```
Epoch 1/5
218/218 ━━━━━━━━━━━━━━━━━━━━ 148s 544ms/step - accuracy: 0.9932 - loss: 0.0216 - val_accuracy: 1.0000 -
Epoch 2/5
218/218 ━━━━━━━━━━━━━━━━━━━━ 107s 449ms/step - accuracy: 0.9955 - loss: 0.0141 - val_accuracy: 1.0000 -
Epoch 3/5
218/218 ━━━━━━━━━━━━━━━━━━━━ 143s 454ms/step - accuracy: 0.9941 - loss: 0.0153 - val_accuracy: 1.0000 -
Epoch 4/5
218/218 ━━━━━━━━━━━━━━━━━━━━ 140s 445ms/step - accuracy: 0.9961 - loss: 0.0120 - val_accuracy: 1.0000 -
Epoch 5/5
218/218 ━━━━━━━━━━━━━━━━━━━━ 100s 448ms/step - accuracy: 0.9954 - loss: 0.0134 - val_accuracy: 1.0000 -
```

```python
train_loss, train_accuracy = model.evaluate(train_generator)
print(f"Train Loss: {train_loss:.4f}\tTrain Accuracy: {train_accuracy * 100:.2f}%")

test_loss, test_accuracy = model.evaluate(test_generator)
print(f"Test Loss: {test_loss:.4f}\tTest Accuracy: {test_accuracy * 100:.2f}%")
```

```
218/218 ━━━━━━━━━━━━━━━━━━━━ 90s 413ms/step - accuracy: 0.9993 - loss: 0.0036
Train Loss: 0.0038      Train Accuracy: 99.90%
33/33 ━━━━━━━━━━━━━━━━━━━━ 5s 138ms/step - accuracy: 1.0000 - loss: 1.2982e-04
Test Loss: 0.0002       Test Accuracy: 100.00%
```

```python
# Save the model in the `.keras` format
model.save('final_MobileNet_model.keras')

# Alternatively, save in the `.h5` format if you prefer
model.save('final_MobileNet_model.h5')
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(m
```

Start coding or generate with AI.