```
import tensorflow as tf
from tensorflow.keras import layers, models
import matplotlib.pyplot as plt

print("TensorFlow version:", tf.__version__)
```

```
# from google.colab import drive
# drive.mount('/content/drive')
```

```
import os, zipfile, pathlib
import tensorflow as tf
from tensorflow.keras import layers
from google.colab import files
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input

IMG_SIZE = (160, 160)
BATCH_SIZE = 32

print("Upload file ")
uploaded = files.upload()

zip_path = next(iter(uploaded))
with zipfile.ZipFile(zip_path, "r") as zip_ref:
    zip_ref.extractall("/content")

data_dir = pathlib.Path("/content/dataset")

print("Data directory:", data_dir)
print("Subfolders:", [p.name for p in data_dir.iterdir() if p.is_dir()])


train_raw = tf.keras.utils.image_dataset_from_directory(
    data_dir,
    validation_split=0.2,
    subset="training",
    seed=123,
    image_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
)

val_raw = tf.keras.utils.image_dataset_from_directory(
    data_dir,
    validation_split=0.2,
    subset="validation",
    seed=123,
    image_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
)


class_names = train_raw.class_names
NUM_CLASSES = len(class_names)
print("Classes:", class_names, "  →  NUM_CLASSES =", NUM_CLASSES)


aug = tf.keras.Sequential([
    tf.keras.layers.RandomFlip("horizontal"),
    tf.keras.layers.RandomRotation(0.05),
    tf.keras.layers.RandomZoom(0.1),
    tf.keras.layers.RandomBrightness(0.1),
])

def aug_then_preprocess(x, y):
    return preprocess_input(aug(x)), y


train_ds = train_raw.map(aug_then_preprocess).cache().prefetch(tf.data.AUTOTUNE)
```

```
train_ds = train_raw.map(aug_then_preprocess).cache().prefetch(tf.data.AUTOTUNE)
val_ds   = val_raw.map(lambda x, y: (preprocess_input(x), y)).cache().prefetch(tf.data.AUTOTUNE)
```

```
import tensorflow as tf
from tensorflow.keras import layers, models

base_model = tf.keras.applications.MobileNetV2(
    input_shape=(160, 160, 3),
    include_top=False,
    weights="imagenet"
)
base_model.trainable = False  # freeze backbone for initial training

# Classification head
model = models.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dropout(0.2),
    layers.Dense(NUM_CLASSES, activation="softmax")
])

# Compile
model.compile(
    optimizer=tf.keras.optimizers.Adam(),
    loss="sparse_categorical_crossentropy",
    metrics=["accuracy"]
)

model.summary()
```

```
history = model.fit(train_ds, validation_data=val_ds, epochs=10)
```

```
# Fine-tune a bit
base_model.trainable = True
for layer in base_model.layers[:-40]:  # unfreeze only top ~40 layers
    layer.trainable = False

model.compile(
    optimizer=tf.keras.optimizers.Adam(1e-5),
    loss="sparse_categorical_crossentropy",
    metrics=["accuracy"]
)
history_ft = model.fit(train_ds, validation_data=val_ds, epochs=5)
```

```
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
import pathlib
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input


image_paths = [p for p in pathlib.Path(data_dir).rglob("*") if p.suffix.lower() in [".jpg", ".jpeg", ".png"]]

print(f"Found {len(image_paths)} images.")
test_path = str(image_paths[2])  # pick one (change index to test different images)
print("Testing on:", test_path)


img = tf.keras.utils.load_img(test_path, target_size=(160, 160))
img_array = tf.keras.utils.img_to_array(img)
img_array = np.expand_dims(img_array, 0)
img_array = preprocess_input(img_array)   # same preprocessing as training
```

```
predictions = model.predict(img_array)
pred_idx = np.argmax(predictions[0])
pred_class = class_names[pred_idx]
confidence = 100 * np.max(predictions[0])


plt.imshow(img)
plt.axis("off")
plt.title(f"Predicted: {pred_class} ({confidence:.2f}%)")
plt.show()
```

```
from google.colab import files
import tensorflow as tf
import matplotlib.pyplot as plt
import numpy as np
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input

#Upload a single image
print("Upload an image you want to classify")
uploaded = files.upload()

#Get the first uploaded file
img_path = next(iter(uploaded))
print("Testing on:", img_path)

#Load & preprocess image (same as training)
img = tf.keras.utils.load_img(img_path, target_size=(160, 160))
img_array = tf.keras.utils.img_to_array(img)
img_array = np.expand_dims(img_array, 0)
img_array = preprocess_input(img_array)

#Predict
predictions = model.predict(img_array)
pred_idx = np.argmax(predictions[0])
pred_class = class_names[pred_idx]
confidence = 100 * np.max(predictions[0])

#Show result
plt.imshow(img)
plt.axis("off")
plt.title(f"Predicted: {pred_class} ({confidence:.2f}%)")
plt.show()
```

```
converter = tf.lite.TFLiteConverter.from_keras_model(model)
converter.optimizations = [tf.lite.Optimize.DEFAULT]
tflite_model = converter.convert()

with open('doorbell_model.tflite', 'wb') as f:
    f.write(tflite_model)
```

```
interpreter = tf.lite.Interpreter(model_path="doorbell_model.tflite")
interpreter.allocate_tensors()
print("TFLite model loaded successfully!")
```