Untitled5

November 18, 2023

[14]:
```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.mobilenet_v2 import MobileNetV2,␣
 ↪preprocess_input, decode_predictions
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

[20]:
```
train_data_dir = 'new/train'
validation_data_dir = 'new/val'
```

[38]:
```
# Use your dataset loading and preprocessing logic here
# For example, using the ImageDataGenerator for data augmentation
datagen = keras.preprocessing.image.ImageDataGenerator(
    preprocessing_function=preprocess_input,
    validation_split=0.5
    )

train_generator = datagen.flow_from_directory(
    train_data_dir,
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    subset='training'
)

validation_generator = datagen.flow_from_directory(
    validation_data_dir,
    target_size=(224, 224),
    batch_size=32,
    class_mode='categorical',
    subset='validation'
)
```

```
Found 10 images belonging to 2 classes.
Found 3 images belonging to 2 classes.
```

[39]:
```
# Create the base model (MobileNetV2)
base_model = tf.keras.applications.MobileNetV2(
    input_shape=(224, 224, 3),
    include_top=False,
    weights='imagenet'
)

# Freeze the layers of the base model
for layer in base_model.layers:
    layer.trainable = False
```

[40]:
```
# Create the custom model on top of the base model
model = keras.Sequential([
    base_model,
    layers.GlobalAveragePooling2D(),
    layers.Dense(1024, activation='relu'),
    layers.Dropout(0.5),
    layers.Dense(2, activation='softmax')  # Two classes: burger and pizza
])
```

[41]:
```
model.compile(optimizer='adam', loss='categorical_crossentropy',␣
 ↪metrics=['accuracy'])
```

[42]:
```
# Train the model
history = model.fit(
    train_generator,
    epochs=10,
    validation_data=validation_generator
)
```

```
Epoch 1/10
1/1 [==============================] - 5s 5s/step - loss: 0.8916 - accuracy:
0.5000 - val_loss: 0.1471 - val_accuracy: 1.0000
Epoch 2/10
1/1 [==============================] - 0s 479ms/step - loss: 0.1426 - accuracy:
1.0000 - val_loss: 0.0081 - val_accuracy: 1.0000
Epoch 3/10
1/1 [==============================] - 0s 488ms/step - loss: 0.0019 - accuracy:
1.0000 - val_loss: 8.1930e-04 - val_accuracy: 1.0000
Epoch 4/10
1/1 [==============================] - 0s 473ms/step - loss: 2.0357e-04 -
accuracy: 1.0000 - val_loss: 1.3668e-04 - val_accuracy: 1.0000
Epoch 5/10
1/1 [==============================] - 0s 482ms/step - loss: 2.2292e-05 -
accuracy: 1.0000 - val_loss: 3.2265e-05 - val_accuracy: 1.0000
Epoch 6/10
1/1 [==============================] - 1s 531ms/step - loss: 5.4478e-06 -
accuracy: 1.0000 - val_loss: 1.0212e-05 - val_accuracy: 1.0000
```

```
Epoch 7/10
1/1 [==============================] - 1s 576ms/step - loss: 7.4385e-06 -
accuracy: 1.0000 - val_loss: 4.0928e-06 - val_accuracy: 1.0000
Epoch 8/10
1/1 [==============================] - 1s 521ms/step - loss: 2.9921e-06 -
accuracy: 1.0000 - val_loss: 1.9471e-06 - val_accuracy: 1.0000
Epoch 9/10
1/1 [==============================] - 1s 551ms/step - loss: 4.8876e-07 -
accuracy: 1.0000 - val_loss: 1.0729e-06 - val_accuracy: 1.0000
Epoch 10/10
1/1 [==============================] - 1s 596ms/step - loss: 8.4638e-07 -
accuracy: 1.0000 - val_loss: 6.3578e-07 - val_accuracy: 1.0000
```

[ ]:
```python
import cv2
import numpy as np
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input

def detect_and_display_object(model, img_path):
    # Load and preprocess the input image
    img = image.load_img(img_path, target_size=(224, 224))
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)
    img_array = preprocess_input(img_array)

    # Make predictions using the model
    predictions = model.predict(img_array)

    # Extract the predicted class label
    class_label = 'burger' if predictions[0][0] > predictions[0][1] else 'pizza'
    confidence = max(predictions[0])

    print(f"Class: {class_label}, Confidence: {confidence}")

    # Load the original image using OpenCV for display
    img_cv2 = cv2.imread(img_path)
    img_cv2 = cv2.cvtColor(img_cv2, cv2.COLOR_BGR2RGB)

    # Draw the bounding box and class label on the image
    font = cv2.FONT_HERSHEY_SIMPLEX
    font_scale = 1
    font_thickness = 2
    box_color = (0, 255, 0)  # Green color for the bounding box

    cv2.putText(img_cv2, f"Class: {class_label}", (10, 30), font, font_scale,␣
↪(255, 255, 255), font_thickness, cv2.LINE_AA)
```

```
    cv2.putText(img_cv2, f"Confidence: {confidence:.2f}", (10, 60), font,␣
 ↪font_scale, (255, 255, 255), font_thickness, cv2.LINE_AA)

    # Display the image with OpenCV
    cv2.imshow('Object Detection', img_cv2)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

# Example usage
random_img_path = 'BI681/test/1.jpg'
detect_and_display_object(model, random_img_path)
```

```
1/1 [==============================] - 0s 72ms/step
Class: pizza, Confidence: 0.999756395816803
```