```
# import os
# os.environ["TF_CPP_MIN_LOG_LEVEL"] = "2"

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt

img_height = 28
img_width = 28
batch_size = 3

# train_generator = datagen.flow_from_directory(
# "/home/nitin/Documents/PYTHON/Python Project/food20dataset/train_set/",
# target_size=(img_height, img_width),
# batch_size=batch_size,
# color_mode="rgb",
# class_mode="sparse",
# shuffle=True,
# subset="training",
# seed=123,
# )
train_ds = tf.keras.utils.image_dataset_from_directory(
'/home/nitin/Documents/PYTHON/Python Project/food20dataset/train_set/',
validation_split=0.1,
color_mode='rgb',
subset="training",
seed=123,
image_size=(img_height, img_width),
batch_size=batch_size,
)
#
# Assuming you have already created the train_ds dataset

# Get the first batch from the dataset
for images, labels in train_ds.take(1):
# Print the shape of the input images
print("Shape of input images:", images.shape)


# validation_generator = datagen.flow_from_directory(
# # "/home/nitin/Documents/PYTHON/Python Project/food20dataset/test_set/",
# 'food20dataset/test_set',
# target_size=(img_height, img_width),
# batch_size=batch_size,
# color_mode="rgb",
# class_mode="sparse",
# shuffle=True,
# subset="validation",
# seed=123,
# )
val_ds = tf.keras.utils.image_dataset_from_directory(
'/home/nitin/Documents/PYTHON/Python Project/food20dataset/test_set/',
validation_split=0.5,
subset="validation",
color_mode='rgb',
seed=123,
image_size=(img_height, img_width),
batch_size=batch_size
)

class_names = train_ds.class_names
print(len(class_names))

normalization_layer = tf.keras.layers.Rescaling(1./255)

AUTOTUNE = tf.data.AUTOTUNE
train_ds = train_ds.cache().prefetch(buffer_size=AUTOTUNE)
val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)

num_classes = 20

# model = tf.keras.Sequential([
# tf.keras.layers.Rescaling(1./255),
# tf.keras.layers.Conv2D(32, 3, activation='relu'),
# tf.keras.layers.MaxPooling2D(),
# tf.keras.layers.Conv2D(64, 3, activation='relu'),
# tf.keras.layers.MaxPooling2D(),
# tf.keras.layers.Conv2D(128, 3, activation='relu'),
# tf.keras.layers.MaxPooling2D(),
# tf.keras.layers.Flatten(),
# tf.keras.layers.Dense(254, activation='relu'),
# tf.keras.layers.Dense(num_classes)

# ])

model = tf.keras.Sequential([
tf.keras.layers.Rescaling(1./255),
tf.keras.layers.Conv2D(32, 3, padding="same", activation='relu'),
tf.keras.layers.MaxPooling2D(),
tf.keras.layers.Conv2D(64, 3, activation='relu'),
tf.keras.layers.MaxPooling2D(),
tf.keras.layers.Conv2D(128, 3, activation='relu'),
tf.keras.layers.MaxPooling2D(),
tf.keras.layers.Flatten(),
tf.keras.layers.Dense(254, activation='relu'),
tf.keras.layers.Dropout(0.5), # Add dropout with a dropout rate of 0.5 (adjust as needed)
tf.keras.layers.Dense(num_classes)
])
```

```
model.compile(
optimizer='adam',
loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
metrics=['accuracy']
)

print(model.summary)


# model.fit(
# train_ds,
# validation_data=val_ds,
# epochs=3
# )
def augment(x, y):
image = tf.image.random_brightness(x, max_delta=0.05)
return image, y


import numpy as np
from sklearn.metrics import classification_report

# Assuming class class_nameslabels are encoded as integers in the range [0, num_classes-1]
true_labels = np.concatenate([y for x, y in val_ds], axis=0)
predictions = model.predict(val_ds).argmax(axis=1)

# Convert integer labels to class names
# class_labels = list(val_ds.class_names)
predicted_labels = [class_names[prediction] for prediction in predictions]
true_labels = [class_names[label] for label in true_labels]

# Generate and print classification report
report = classification_report(true_labels, predicted_labels, target_names=class_names)
print(report)
```