Food-101 Image Classification in Google Colab — Full Explanation

```
#STEP 1: Install TensorFlow Datasets
#!pip install tensorflow-datasets

'''What this does:
We install tensorflow-datasets, a library that provides many pre-built datasets like Food-101, MNIST, CIFAR-10, etc., ready for use in ML mc
```

```
'What this does:\nWe install tensorflow-datasets, a library that provides many pre-built datasets like Food-101, MNIST, CIFAR-10, etc.,
```

```
#STEP 2: Import Required Libraries
import tensorflow as tf
import tensorflow_datasets as tfds
import matplotlib.pyplot as plt
import numpy as np
```

What this does:

- tensorflow: Deep learning framework (we'll use it to build/train our model)
- tensorflow_datasets (tfds): To load the Food-101 dataset
- matplotlib.pyplot: For image plotting
- numpy: For numerical operations and predictions

```
#STEP 3: Load the Food-101 Dataset
(ds_train, ds_test), ds_info = tfds.load(
    'food101',
    split=['train', 'validation'],
    shuffle_files=True,
    as_supervised=True,
    with_info=True
)
```

```
WARNING:absl:Variant folder /root/tensorflow_datasets/food101/2.0.0 has no dataset_info.json
Downloading and preparing dataset Unknown size (download: Unknown size, generated: Unknown size, total: Unknown size) to /root/tensorflc
 Dl Completed...: 100%        1/1 [08:40<00:00, 187.39s/ url]
 Dl Size...: 100%       4764/4764 [08:40<00:00, 26.60 MiB/s]
 Extraction completed...: 100%       101008/101008 [08:39<00:00, 1236.01 file/s]
Dataset food101 downloaded and prepared to /root/tensorflow_datasets/food101/2.0.0. Subsequent calls will reuse this data.
```

What this does step 3:

- Loads the Food-101 dataset.
- split=['train', 'validation']: Fetches training and test data.
- as_supervised=True: Each sample is (image, label)
- with_info=True: Returns metadata (like label names, shapes, etc.)

```
#STEP 4: View Dataset Information
print("Total Classes:", ds_info.features['label'].num_classes)
print("Sample Classes:", ds_info.features['label'].names[:10])
```

```
Total Classes: 101
Sample Classes: ['apple_pie', 'baby_back_ribs', 'baklava', 'beef_carpaccio', 'beef_tartare', 'beet_salad', 'beignets', 'bibimbap', 'brea
```

What this does step 4

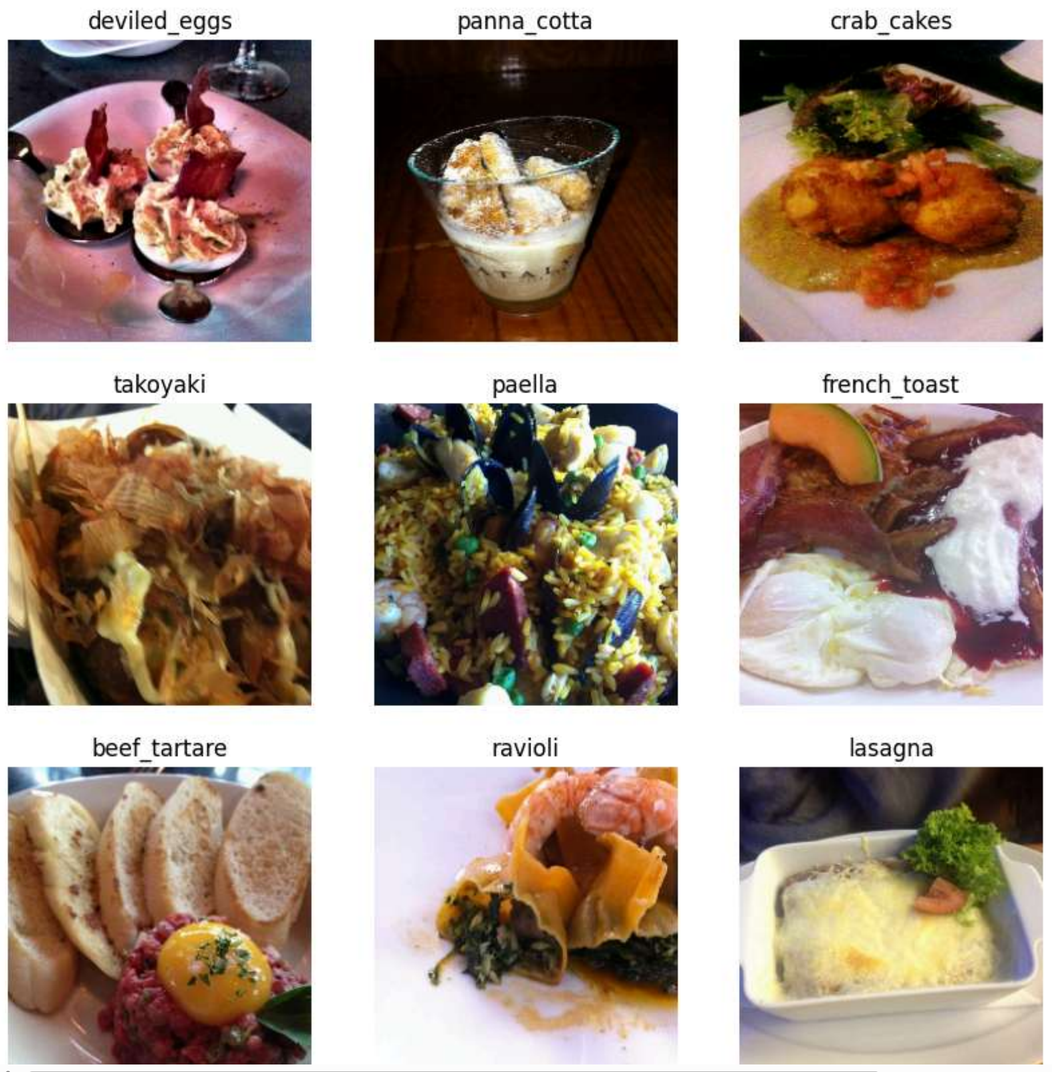
- ds_info: Metadata about the dataset
- Prints number of food classes and some class names like "pizza", "sushi", etc.

```
#STEP 5: Preprocess the Images
IMG_SIZE = 224
BATCH_SIZE = 32

def preprocess(image, label):
    image = tf.image.resize(image, (IMG_SIZE, IMG_SIZE))  # Resize to 224x224
    image = tf.cast(image, tf.float32) / 255.0             # Normalize pixels [0,1]
    return image, label

# Apply preprocessing to the datasets
ds_train = ds_train.map(preprocess).batch(BATCH_SIZE).prefetch(tf.data.AUTOTUNE)
ds_test = ds_test.map(preprocess).batch(BATCH_SIZE).prefetch(tf.data.AUTOTUNE)
```

In step 5

- Resizes images to 224x224 — standard input size for CNNs like MobileNet.
- Normalizes pixel values (0–255 → 0–1).
- Batching: Combines samples into groups of 32 for faster training.
- Prefetching: Loads next batch while current is training (for speed).

```
#STEP 6: Visualize Images
for images, labels in ds_train.take(1):
    plt.figure(figsize=(10, 10))
    for i in range(9):
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(images[i].numpy())
        label = ds_info.features['label'].int2str(labels[i].numpy())
        plt.title(label)
        plt.axis("off")
    plt.show()
```



in step 6

- Displays 9 food images from the training set.
- Converts label integers to readable food names (e.g., "pizza").

```
#STEP 7: Build the CNN Model (Transfer Learning)
base_model = tf.keras.applications.MobileNetV2(
    input_shape=(224, 224, 3),
    include_top=False,
    weights='imagenet'
)

base_model.trainable = False  # Freeze base layers

model = tf.keras.Sequential([
    base_model,
    tf.keras.layers.GlobalAveragePooling2D(),       # Flatten feature maps
    tf.keras.layers.Dense(101, activation='softmax')  # 101 food categories
])
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_weights_tf_dim_ordering_tf_
9406464/9406464 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

What In Step 7:

- Loads MobileNetV2 pretrained on ImageNet (already knows general image features).
- Freezes it to use as a feature extractor.
- Adds a custom classification layer (Dense(101)) for our 101 food classes.
- softmax: Returns probabilities for each class.

```
#STEP 8: Compile the Model
model.compile(
    optimizer='adam',
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)
```

What in step 8:

- adam: Efficient optimizer for training.
- sparse_categorical_crossentropy: Used for integer-labeled multi-class classification.
- accuracy: Evaluation metric.

```
#STEP 9: Train the Model
history = model.fit(ds_train, epochs=5, validation_data=ds_test)
```

```
Epoch 1/5
2368/2368 ━━━━━━━━━━━━━━━━━━━━ 4231s 2s/step - accuracy: 0.3781 - loss: 2.6152 - val_accuracy: 0.5701 - val_loss: 1.6458
Epoch 2/5
2368/2368 ━━━━━━━━━━━━━━━━━━━━ 4281s 2s/step - accuracy: 0.5732 - loss: 1.6621 - val_accuracy: 0.5886 - val_loss: 1.5829
Epoch 3/5
2368/2368 ━━━━━━━━━━━━━━━━━━━━ 4243s 2s/step - accuracy: 0.6194 - loss: 1.4747 - val_accuracy: 0.5901 - val_loss: 1.5798
Epoch 4/5
2368/2368 ━━━━━━━━━━━━━━━━━━━━ 4226s 2s/step - accuracy: 0.6426 - loss: 1.3604 - val_accuracy: 0.5869 - val_loss: 1.6080
Epoch 5/5
2368/2368 ━━━━━━━━━━━━━━━━━━━━ 4210s 2s/step - accuracy: 0.6598 - loss: 1.2892 - val_accuracy: 0.5903 - val_loss: 1.6178
```

What in STEP 9:

- Trains the model for 5 epochs (5 passes over training data).
- Validates performance on the test set after each epoch.

```
#STEP 10: Evaluate the Model
loss, accuracy = model.evaluate(ds_test)
print(f"Test Accuracy: {accuracy * 100:.2f}%")
```

```
790/790 ━━━━━━━━━━━━━━━━━━━━ 1061s 1s/step - accuracy: 0.5914 - loss: 1.6161
Test Accuracy: 59.03%
```

What in step 10:

- Tests final model on unseen data.
- Prints accuracy — how well it classifies real food images.

```
#STEP 11: Make Predictions & Show Results
class_names = ds_info.features['label'].names

for images, labels in ds_test.take(1):
    image = images[0]
    true_label = class_names[labels[0].numpy()]

    prediction = model.predict(tf.expand_dims(image, axis=0))
    predicted_label = class_names[np.argmax(prediction)]

    plt.imshow(image)
    plt.title(f"Predicted: {predicted_label}\nTrue: {true_label}")
    plt.axis('off')
    plt.show()
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 2s 2s/step
```

What in step 11

- Selects 1 image from test data.
- Predicts its label using the model.
- Displays the image with its true and predicted labels.

```
#STEP 12: Save the Trained Model (Optional)
model.save("food101_model.h5")  # Save to file
#Saves the trained model in .h5 format so you can reuse it later.
```

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

All Step's Code Block Purpose

1) Install Datasets

2)Import Libraries

3) Load Food-101

4) View Class Info

5) Preprocess Images

6) Visualize Samples

7) Build CNN Model