In [1]:

```
import tensorflow as tf
```

In [2]:

```
data_dir = "data/food-101/images/"
```

In [68]:

```
from tensorflow.keras import mixed_precision

mixed_precision.set_global_policy('float32')
```

In [64]:

```
from tensorflow.keras.utils import image_dataset_from_directory

all_images = image_dataset_from_directory(directory=data_dir,
                                          label_mode="categorical",
                                          batch_size=32,
                                          image_size=(256, 256)
                                          )
```

```
Found 101000 files belonging to 101 classes.
```

In [65]:

```
len(all_images.class_names)
```

Out[65]:

```
101
```

In [69]:

```
from tensorflow.keras import Sequential, Model, Input
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.keras.applications import EfficientNetB0

data_aug =  Sequential([
    tf.keras.layers.RandomRotation(0.2),
    tf.keras.layers.RandomZoom(0.2),
    tf.keras.layers.RandomHeight(0.2),
    tf.keras.layers.RandomWidth(0.2),
#     tf.keras.layers.Rescaling(1.0/255) # But not for the Efficient layer as its own Rescaling Layer

], name ="Data_Augmentation_Layer")

base_model = EfficientNetB0(include_top=False)
for i in base_model.layers[:-10]:
    i.trainable = False

ip = Input(shape=(256, 256, 3), name="Input_layer")
x = data_aug(ip)
x = base_model(x, training=False)
x = GlobalAveragePooling2D(name="Pooling_layer")(x)
op = Dense(101, activation="softmax", name="output_layer")(x)

food_vision_model = Model(ip, op)
```

In [70]:

```
food_vision_model.summary()
```

```
Model: "model_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 Input_layer (InputLayer)    [(None, 256, 256, 3)]     0

 Data_Augmentation_Layer (Se  (None, 256, 256, 3)      0
 quential)

 efficientnetb0 (Functional)  (None, None, None, 1280)  4049571

 Pooling_layer (GlobalAverag  (None, 1280)             0
 ePooling2D)

 output_layer (Dense)        (None, 101)               129381

=================================================================
Total params: 4,178,952
Trainable params: 1,022,613
Non-trainable params: 3,156,339
_________________________________________________________________
```

In [71]:

```
food_vision_model.compile(loss="categorical_crossentropy",
                          optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
                          metrics=["accuracy"]
                          )
```

In [72]:

```
# Creating a model callBack Function
from tensorflow.keras.callbacks import ModelCheckpoint

checkpoint_dir = "models/food_vision_model/"

checkpoint_callback = ModelCheckpoint(filepath=checkpoint_dir,
                                      save_best_only=True,
                                      save_weights_only=False,
                                      save_freq="epoch",
                                      monitor='loss')
```

In [74]:

```
food_vision_model.fit(all_images,
                      epochs=30,
                      steps_per_epoch=len(all_images),
                      callbacks=[checkpoint_callback])
```

```
Epoch 1/30
3157/3157 [==============================] - ETA: 0s - loss: 1.9760 - accuracy: 0.5078

WARNING:absl:Function `_wrapped_model` contains input name(s) Input_layer with unsupported characters which will
be renamed to input_layer in the SavedModel.
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_c
ompiled_convolution_op, _jit_compiled_convolution_op, _jit_compiled_convolution_op while saving (showing 5 of 8
1). These functions will not be directly callable after loading.

INFO:tensorflow:Assets written to: models/food_vision_model\assets

INFO:tensorflow:Assets written to: models/food_vision_model\assets

3157/3157 [==============================] - 1210s 382ms/step - loss: 1.9760 - accuracy: 0.5078
Epoch 2/30
3157/3157 [==============================] - ETA: 0s - loss: 1.4166 - accuracy: 0.6315

WARNING:absl:Function `_wrapped_model` contains input name(s) Input_layer with unsupported characters which will
be renamed to input_layer in the SavedModel.
WARNING:absl:Found untraced functions such as _jit_compiled_convolution_op, _jit_compiled_convolution_op, _jit_c
```

In [76]:

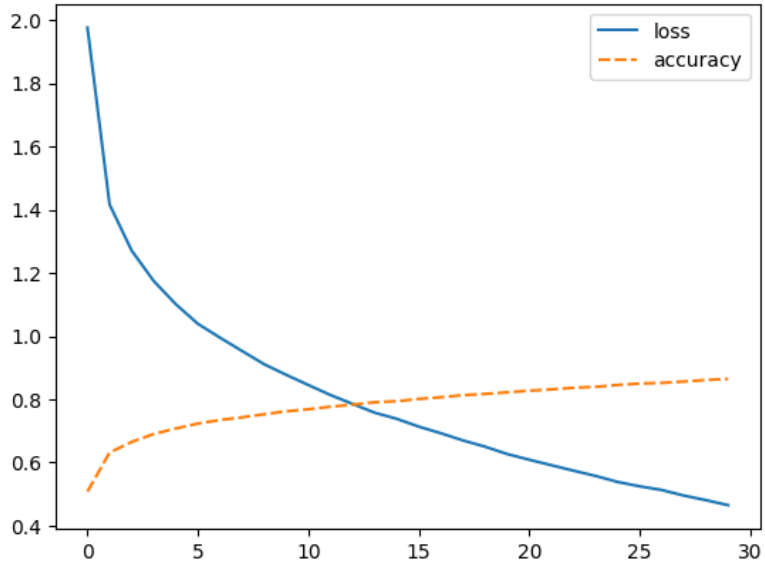
```
food_vision_model.save("models/food_vison_model.h5")
```

In [87]:

```
import pandas as pd
import seaborn as sns


df = pd.DataFrame(food_vision_model.history.history)
sns.lineplot(df)
```

Out[87]:

```
<AxesSubplot: >
```



In [88]:

```
test_dir = "data/food-101/test/"
```

In [90]:

```
test_images = image_dataset_from_directory(directory=test_dir,
                                           label_mode="categorical",
                                           batch_size=32,
                                           image_size=(256, 256)
                                           )
```

```
Found 25250 files belonging to 101 classes.
```

In [107]:

```
y_labels = []
for images, labels in test_images.unbatch(): # unbatch the test data and get images and labels
    y_labels.append(labels.numpy().argmax()) # append the index which has the largest value (labels are one-hot)
y_labels[:10] # check what they look like (unshuffled)
```

Out[107]:

```
[48, 55, 5, 11, 97, 84, 22, 72, 59, 49]
```

In [92]:

```
y_preds = food_vision_model.predict(test_images)
```

```
790/790 [==============================] - 428s 543ms/step
```

In [94]:

```
y_classes = y_preds.argmax(axis=1)
```

In [95]:

```
y_classes
```

Out[95]:

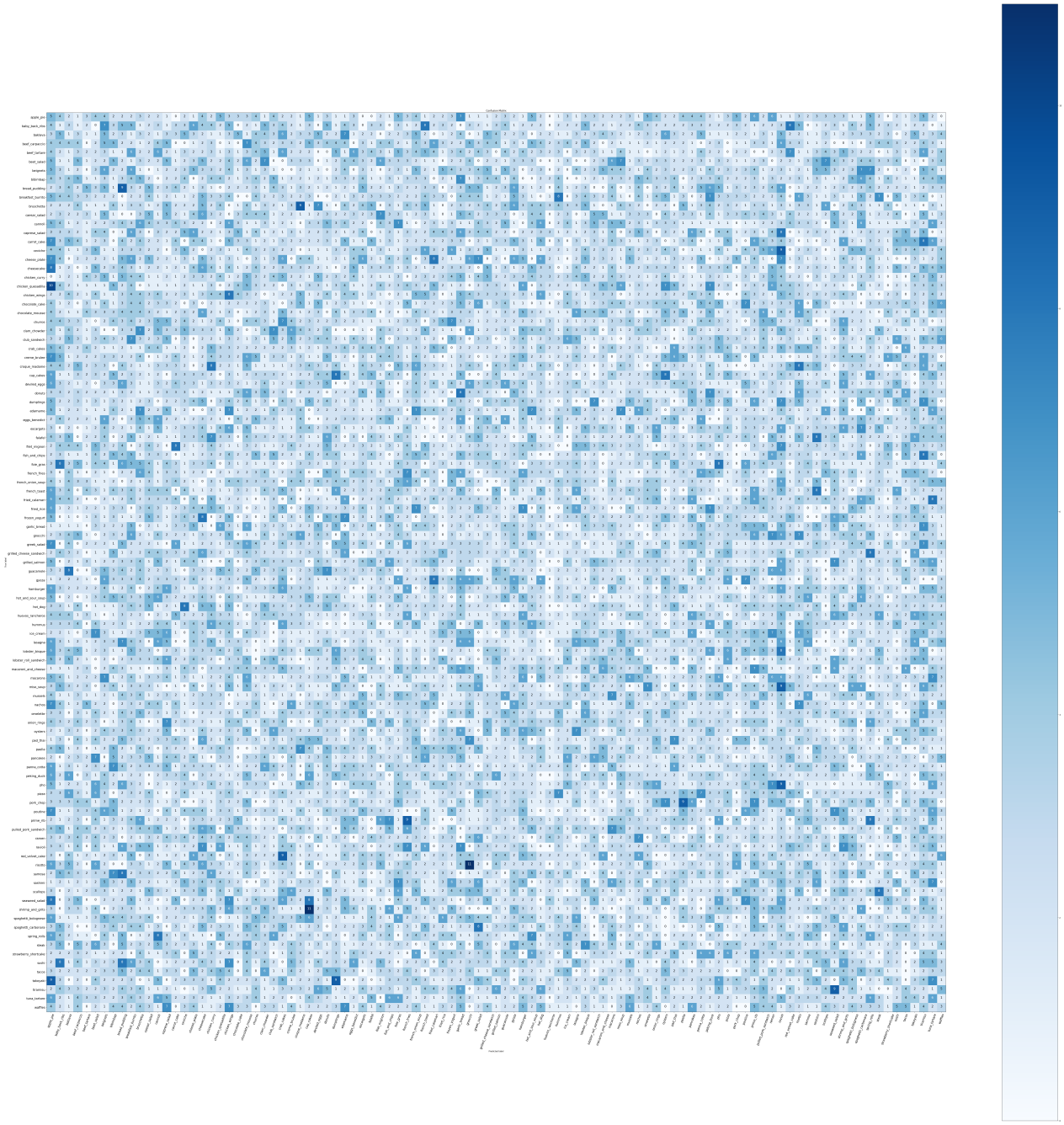
```
array([51, 18, 21, ..., 36, 20, 73], dtype=int64)
```

In [129]:

```
import itertools
import matplotlib.pyplot as plt
import numpy as np
from sklearn.metrics import confusion_matrix

def make_confusion_matrix(y_true, y_pred, classes=None, figsize=(100, 100), text_size=15, norm=False, savefig=False):
    cm = confusion_matrix(y_true, y_pred)
    cm_norm = cm.astype("float") / cm.sum(axis=1)[:, np.newaxis] # normalize it
    n_classes = cm.shape[0] # find the number of classes we're dealing with

    fig, ax = plt.subplots(figsize=figsize)
    cax = ax.matshow(cm, cmap=plt.cm.Blues) # colors will represent how 'correct' a class is, darker == better
    fig.colorbar(cax)

    if classes:
        labels = classes
    else:
        labels = np.arange(cm.shape[0])

  # Label the axes
    ax.set(title="Confusion Matrix",
         xlabel="Predicted label",
         ylabel="True label",
         xticks=np.arange(n_classes),
         yticks=np.arange(n_classes),
         xticklabels=labels,
         yticklabels=labels)

    ax.xaxis.set_label_position("bottom")
    ax.xaxis.tick_bottom()

    plt.xticks(rotation=70, fontsize=text_size)
    plt.yticks(fontsize=text_size)

    threshold = (cm.max() + cm.min()) / 2.

    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        if norm:
            plt.text(j, i, f"{cm[i, j]} ({cm_norm[i, j]*100:.1f}%)",
              horizontalalignment="center",
              color="white" if cm[i, j] > threshold else "black",
              size=text_size)
        else:
            plt.text(j, i, f"{cm[i, j]}",
              horizontalalignment="center",
              color="white" if cm[i, j] > threshold else "black",
              size=text_size)

    if savefig:

        fig.savefig("confusion_matrix.png")
```

In [130]:

```
make_confusion_matrix(y_labels, y_classes, test_images.class_names, savefig=True)
```



In [131]:

```
food_vision_model.evaluate(test_images)
```

```
790/790 [==============================] - 83s 102ms/step - loss: 0.3803 - accuracy: 0.8827
```

Out[131]:

```
[0.38031995296478271, 0.88265347480773931]
```

In [139]:

```
def load_and_prep_image(filename, img_shape=256, scale=True):
  img = tf.io.read_file(filename)
  img = tf.io.decode_image(img)
  img = tf.image.resize(img, [img_shape, img_shape])
  if scale:
    return img/255.
  else:
    return img
```

In [175]:

```
class_names = test_images.class_names
import os
import random

plt.figure(figsize=(17, 10))

for i in range(3):
    class_name = random.choice(class_names)
    filename = random.choice(os.listdir(data_dir + "/" + class_name))
    filepath = data_dir + class_name + "/" + filename

    img = load_and_prep_image(filepath, scale=False)
    pred_prob = food_vision_model.predict(tf.expand_dims(img, axis=0))
    pred_class = class_names[pred_prob.argmax()]

    plt.subplot(1, 3, i+1)
    plt.imshow(img/255.)
    if class_name == pred_class:
        title_color = "g"
    else:
        title_color = "r"
    plt.title(f"actual: {class_name}, pred: {pred_class}, prob: {pred_prob.max():.2f}", c=title_color)
    plt.axis(False);
```

```
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 25ms/step
```

In [ ]: