A project report on

Food Vision

Submitted in partial fulfilment for the award of the degree of

B. Tech CSE (AI and ML)

by

D L K Trinadh 21BAI1579

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

April 2024

Abstract

Food Vision is a crucial task in the domain of computer vision, holding significant implications for dietary monitoring, food logging applications, and restaurant menu analysis. This project introduces an innovative approach to address this challenge by leveraging machine vision techniques and deep learning algorithms.

The proposed system employs convolutional neural networks (CNNs) and advanced image processing methods to analyze food images and automatically identify individual food items present in the scene. By training on a comprehensive dataset of annotated food images encompassing diverse cuisines and dishes, the system learns to accurately classify various food items in real-time.

A multi-step pipeline is devised for preprocessing food images and extracting pertinent features crucial for food item recognition. The architecture of the deep learning model is meticulously designed to effectively capture both visual features and contextual information inherent in food images, enabling robust and accurate identification of food items.

Experimental results demonstrate the efficacy of the proposed approach in achieving high accuracy and efficiency in food item recognition tasks. The potential applications of this system extend to dietary monitoring applications, personalized nutrition recommendations, and restaurant menu analysis, facilitating informed decision-making and promoting healthier eating habits.

Overall, this project underscores the significant potential of machine vision and deep learning techniques in automating food item recognition, thereby contributing to advancements in dietary analysis and food-related applications across various domains.

Table of contents

Introduction

In the era characterized by digital transformation and heightened emphasis on health and nutrition, the automated recognition of food items stands as a pivotal pursuit within the realm of computer vision. With the ubiquity of smartphones and the proliferation of food-related applications, the demand for automated tools to facilitate dietary assessment, meal planning, and restaurant menu analysis has never been more pronounced.

Traditional methods for food item recognition often rely on manual feature engineering and handcrafted descriptors, which may falter in capturing the nuanced visual characteristics inherent in food images. These approaches are susceptible to challenges such as variations in lighting, food presentation styles, and occlusions, hindering their efficacy in real-world settings.

The recent advancements in deep learning have heralded a paradigm shift in computer vision, enabling the automatic extraction of high-level features directly from raw image data. Convolutional neural networks (CNNs) have emerged as powerful tools for capturing spatial information in images, while recurrent neural networks (RNNs) excel at modeling temporal dependencies.

In this project, we embark on a journey to develop an innovative framework for automated food item recognition harnessing the power of machine vision and deep learning techniques. Our endeavor aims to bridge the gap between cutting-edge research and practical applications, with the ultimate goal of empowering individuals to make informed dietary choices and fostering healthier eating habits.

Drawing inspiration from recent successes in human activity recognition and object detection, we propose a multi-stage pipeline that preprocesses food images, extracts discriminative features, and leverages a hybrid CNN-RNN architecture to classify food items with high accuracy and efficiency. Through rigorous experimentation and evaluation on diverse food datasets, we aim to demonstrate the efficacy and versatility of our proposed system in addressing the challenges inherent in food item recognition tasks.

Objective

The primary objective of this research is to develop an automated food item recognition system leveraging machine vision and deep learning techniques. The proposed system aims to accurately and efficiently identify a diverse array of food items from images, facilitating dietary assessment, meal planning, and restaurant menu analysis.

1. Investigate and elucidate fundamental concepts of automated food item recognition, including image preprocessing methods, feature extraction techniques, and classification algorithms.

2. Evaluate machine vision and deep learning approaches utilized in automated food item recognition, analyzing their efficacy in capturing visual characteristics and discriminating between food categories.

3. Address challenges inherent in food image analysis, such as variations in food presentation, occlusions, and scale, and explore strategies to mitigate these challenges through data augmentation and model optimization.

4. Explore emerging trends and future research directions in automated food item recognition, including the integration of contextual information, multi-modal fusion techniques, and real-time deployment in mobile applications.

5. Provide practical insights and recommendations for researchers and practitioners seeking to develop effective automated food item recognition systems, with the overarching goal of advancing machine vision technology in the domain of dietary assessment and nutrition monitoring.

Problem statement

Automated food item recognition is a critical challenge within the domain of computer vision, with significant implications for dietary monitoring, nutrition analysis, and restaurant menu understanding. Accurately identifying and categorizing food items from images presents several challenges:

1. **Varied Visual Characteristics:** Food items exhibit diverse visual attributes, including shape, color, texture, and presentation style. Effectively capturing and encoding these characteristics to enable accurate classification poses a fundamental challenge.

2. **Occlusions and Variations:** Food images captured in real-world settings often suffer from occlusions, varying lighting conditions, and background clutter. These factors can obscure important visual cues and hinder the accurate recognition of food items.

3. **Scale and Perspective:** Food items may appear at different scales and perspectives within images, further complicating the recognition task. Robust algorithms capable of handling variations in scale and viewpoint are essential for accurate food item recognition.

4. **Real-Time Processing:** In many applications, such as dietary monitoring apps and restaurant menu analysis tools, food item recognition needs to be performed in real-time. This necessitates the development of efficient algorithms and models capable of processing image data quickly without compromising accuracy.

5. **Generalization Across Cuisines:** Food recognition systems must demonstrate the ability to generalize across diverse cuisines and cultural contexts. This requires robust models trained on comprehensive datasets encompassing a wide range of food categories and culinary traditions.

Dataset used

The dataset used in this project “Food Vision” is 10_food_classes_10_percent

And the “10_food_classes_10_percent” contains:

```
There are 2 directories and 0 images in '10_food_classes_10_percent'.
There are 10 directories and 0 images in '10_food_classes_10_percent/train'.
There are 0 directories and 75 images in '10_food_classes_10_percent/train/sushi'.
There are 0 directories and 75 images in '10_food_classes_10_percent/train/hamburger'.
There are 0 directories and 75 images in '10_food_classes_10_percent/train/chicken_curry'.
There are 0 directories and 75 images in '10_food_classes_10_percent/train/grilled_salmon'.
There are 0 directories and 75 images in '10_food_classes_10_percent/train/steak'.
There are 0 directories and 75 images in '10_food_classes_10_percent/train/chicken_wings'.
There are 0 directories and 75 images in '10_food_classes_10_percent/train/pizza'.
There are 0 directories and 75 images in '10_food_classes_10_percent/train/ramen'.
There are 0 directories and 75 images in '10_food_classes_10_percent/train/ice_cream'.
There are 0 directories and 75 images in '10_food_classes_10_percent/train/fried_rice'.
There are 10 directories and 0 images in '10_food_classes_10_percent/test'.
There are 0 directories and 250 images in '10_food_classes_10_percent/test/sushi'.
There are 0 directories and 250 images in '10_food_classes_10_percent/test/hamburger'.
There are 0 directories and 250 images in '10_food_classes_10_percent/test/chicken_curry'.
There are 0 directories and 250 images in '10_food_classes_10_percent/test/grilled_salmon'.
There are 0 directories and 250 images in '10_food_classes_10_percent/test/steak'.
There are 0 directories and 250 images in '10_food_classes_10_percent/test/chicken_wings'.
There are 0 directories and 250 images in '10_food_classes_10_percent/test/pizza'.
There are 0 directories and 250 images in '10_food_classes_10_percent/test/ramen'.
There are 0 directories and 250 images in '10_food_classes_10_percent/test/ice_cream'.
There are 0 directories and 250 images in '10_food_classes_10_percent/test/fried_rice'.
```

- This dataset contains a total of 10 types of food images in both train data and test data.
- The train data contains 75 images of each 10 food types whereas test data contains 250 images of each 10 food types.

Proposed work

The proposed work for the implementation of the automated food item recognition project can be as follows:

1. **Data Collection and Preprocessing:**
    - Acquire a diverse dataset of food images from publicly available sources or curated datasets, ensuring coverage of various cuisines and food categories.
    - Preprocess the food images to standardize their size, color, and orientation, and apply techniques such as data augmentation to enhance the diversity and quantity of training data.
2. **Feature Extraction and Representation Learning:**
    - Extract relevant visual features from the food images using deep learning-based feature extraction methods, such as pre-trained convolutional neural networks (CNNs) like ResNet, VGG, or Inception.
    - Investigate domain-specific feature representations that capture discriminative characteristics of food items, such as texture, shape, and color, using techniques like transfer learning and fine-tuning.
3. **Model Architecture Design:**
    - Design a deep learning architecture tailored for food item recognition, leveraging the power of CNNs for spatial feature extraction and temporal information encoding.
    - Explore various architectures, including single-stage CNNs, multi-stage CNNs with attention mechanisms, and hybrid architectures combining CNNs with recurrent neural networks (RNNs) for capturing temporal dependencies in sequential food images.

4. **Model Training and Optimization:**

    - Train the proposed deep learning model on the preprocessed food image dataset, optimizing hyperparameters such as learning rate, batch size, and regularization techniques.
    - Employ techniques like early stopping, learning rate scheduling, and dropout regularization to prevent overfitting and enhance model generalization capabilities.

5. **Evaluation and Benchmarking:**

    - Evaluate the performance of the automated food item recognition system on held-out validation and test datasets using standard evaluation metrics such as accuracy, precision, recall, and F1-score.
    - Conduct comparative analysis against existing food recognition methods and benchmark datasets to assess the system's effectiveness and robustness across different food categories and environmental conditions.

6. **Deployment and Integration:**

    - Explore deployment strategies for integrating the automated food item recognition system into practical applications, such as mobile apps, dietary monitoring platforms, and restaurant management systems.
    - Investigate scalability and computational efficiency considerations to ensure the system's feasibility for real-time inference and large-scale deployment in diverse settings.

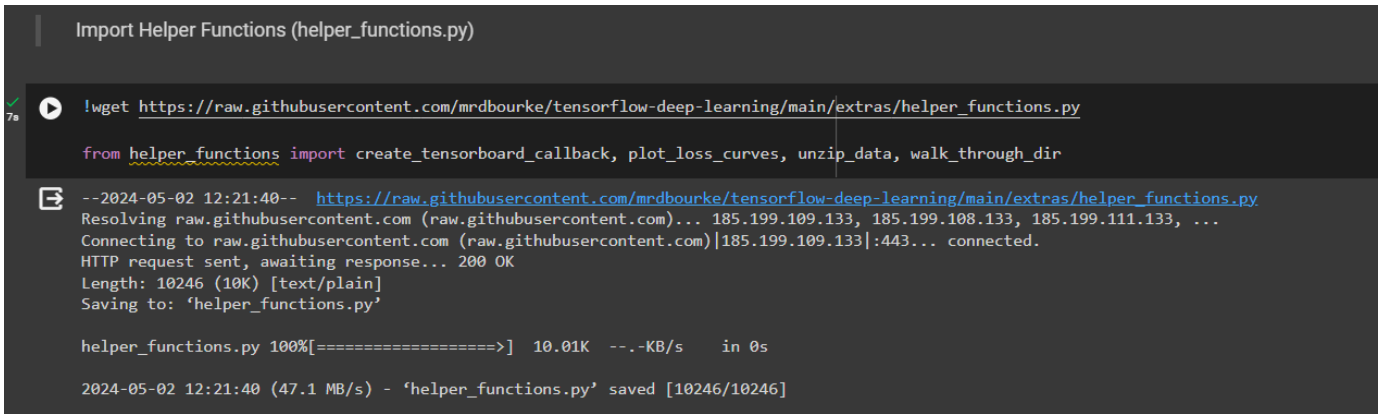
7. **Documentation and Knowledge Sharing:**

    - Document the development process, experimental findings, and insights gained from the project for knowledge dissemination and future reference.
    - Share the project codebase, trained models, and evaluation results with the research community through open-access repositories and publications to facilitate reproducibility and collaboration.

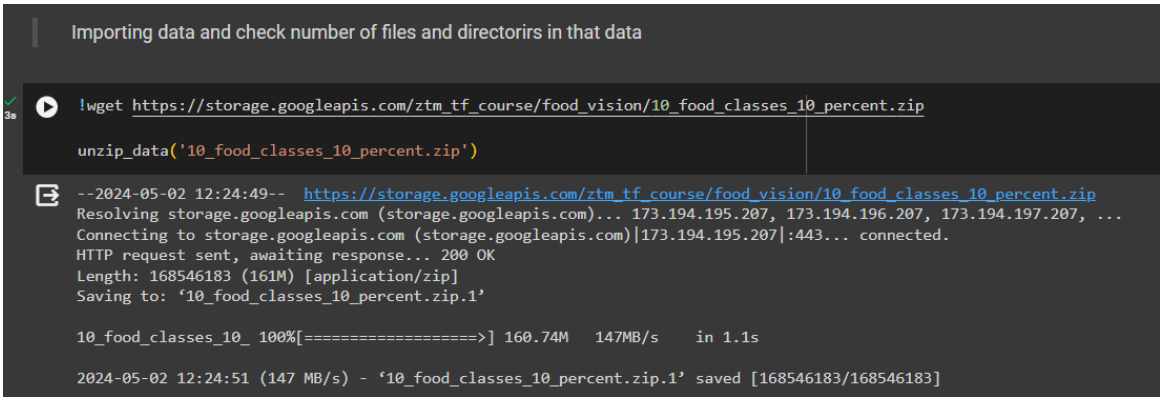
Modules explanation

Installing dependencies and loading libraries

The required helper functions are being imported.



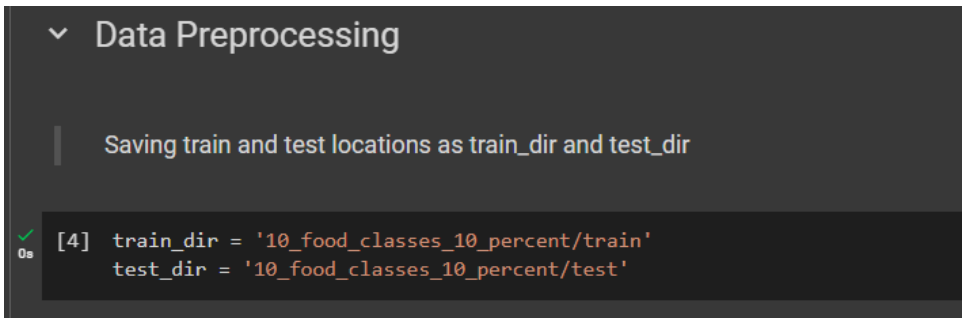
Loading the dataset



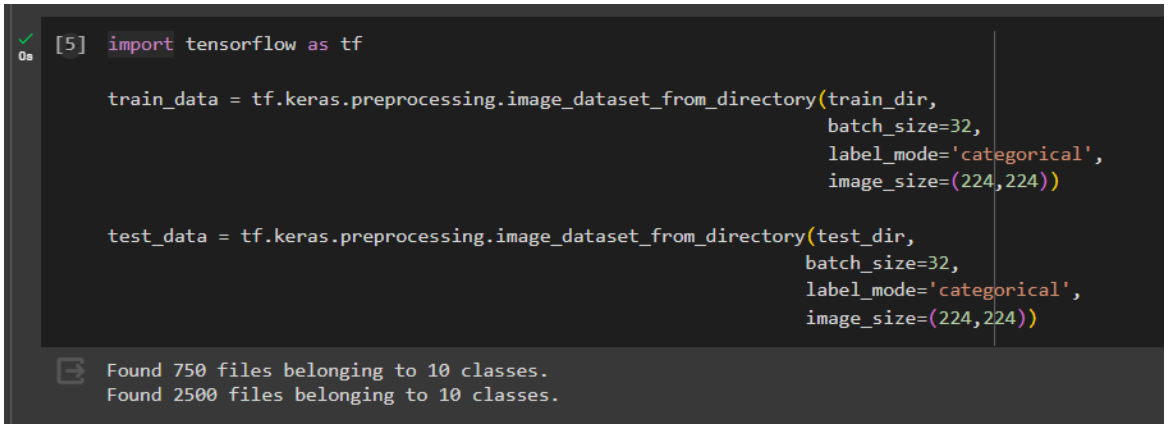
Data Pre-processing

The dataset consists of train,csv and test.csv.

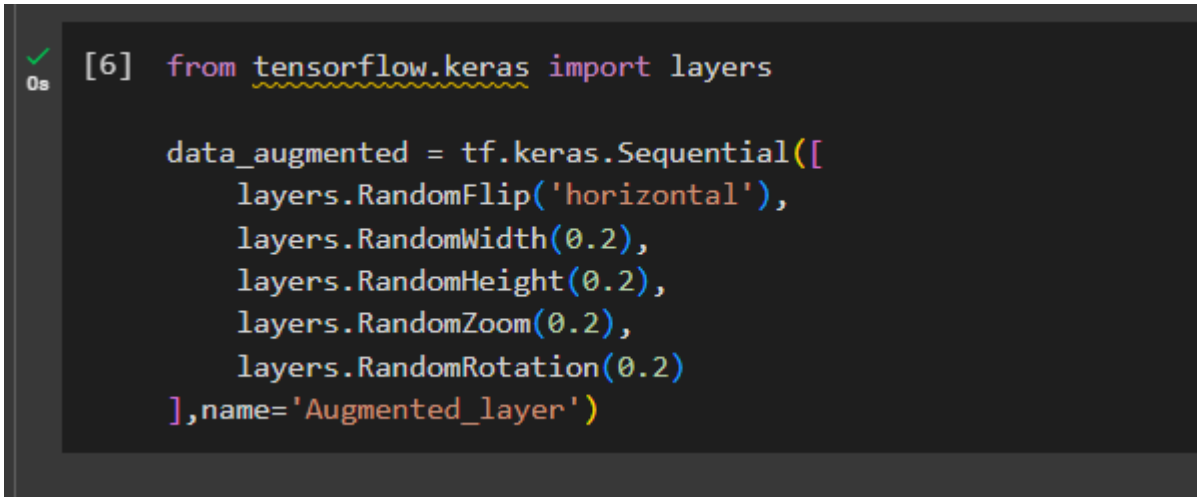
Both the files are loaded.



Now extract the files and divide it into train_data and test_data.



Now create an Augmented layer for the input images.



Feature Extraction Model Building

Functional Model using EfficientNetB0 model already prepared with layers

```
[7]  import tensorflow as tf
     from tensorflow.keras import layers

     base_model = tf.keras.applications.EfficientNetB0(include_top=False)
     base_model.trainable = False

     inputs = layers.Input(shape=(224,224,3), name='Input_layer')
     x = data_augmented(inputs)
     x = base_model(x, training=False)
     x = layers.GlobalAveragePooling2D()(x)
     outputs = layers.Dense(10, activation='softmax')(x)

     model_1 = tf.keras.Model(inputs, outputs)

     model_1.compile(loss='categorical_crossentropy', metrics=['accuracy'], optimizer='adam')
```

Now Train the model for only 25% of data.

And calculate train_accuracy, train_loss, val_accuracy, and val_loss for these each 5 epochs.

Now save the model in “history_1”.

```
[9]  initial_epochs = 5

     history_1 = model_1.fit(train_data,
                             epochs=5,
                             steps_per_epoch=len(train_data),
                             validation_data=test_data,
                             validation_steps=int(0.25*len(test_data)))

Epoch 1/5
24/24 [==============================] - 148s 6s/step - loss: 1.9686 - accuracy: 0.3507 - val_loss: 1.4102 - val_accuracy: 0.7138
Epoch 2/5
24/24 [==============================] - 139s 6s/step - loss: 1.3007 - accuracy: 0.7053 - val_loss: 1.0083 - val_accuracy: 0.7911
Epoch 3/5
24/24 [==============================] - 139s 6s/step - loss: 0.9962 - accuracy: 0.7667 - val_loss: 0.7955 - val_accuracy: 0.8273
Epoch 4/5
24/24 [==============================] - 140s 6s/step - loss: 0.8305 - accuracy: 0.7773 - val_loss: 0.6880 - val_accuracy: 0.8421
Epoch 5/5
24/24 [==============================] - 102s 4s/step - loss: 0.7558 - accuracy: 0.8133 - val_loss: 0.6183 - val_accuracy: 0.8569
```

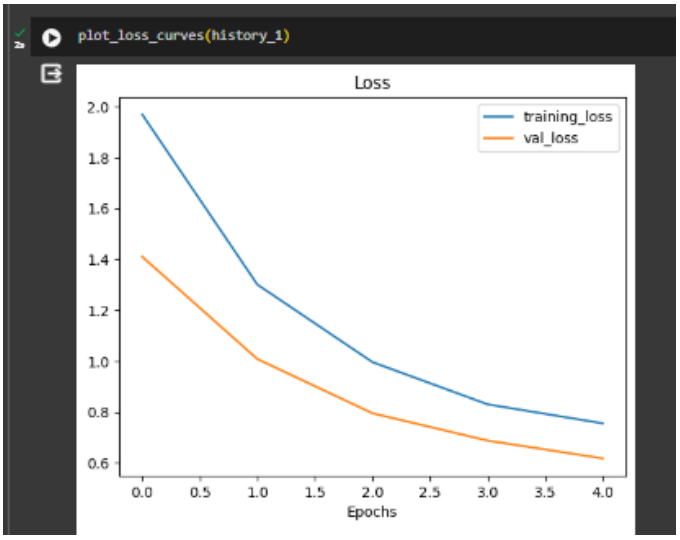
Now train the 100% of data and calculate train_loss, train_accuracy.

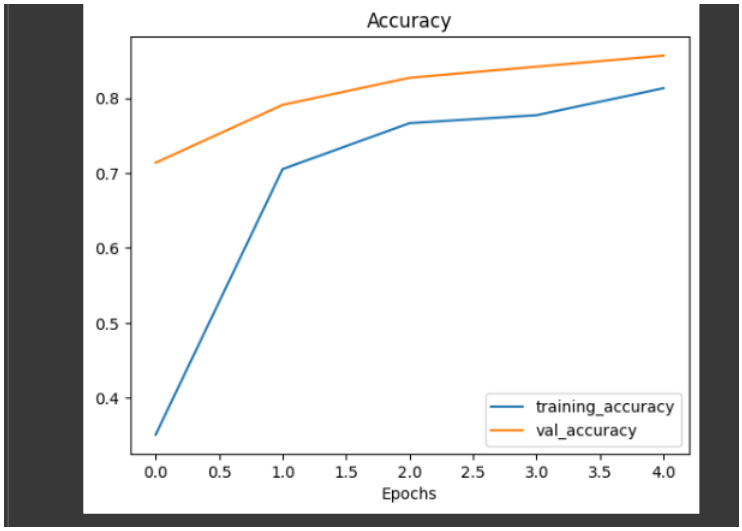
```
[10] feature_results = model_1.evaluate(test_data)

79/79 [==============================] - 200s 3s/step - loss: 0.6225 - accuracy: 0.8416
```

Now plot the loss_curves for the “history_1”.

```
plot_loss_curves(history_1)
```





Check the number of layers in the base_model which are trainable.

```
[14] for layer_number,layer in enumerate(base_model.layers):
        print(layer_number, layer.name, layer.trainable)

0 input_1 False
1 rescaling False
2 normalization False
3 rescaling_1 False
4 stem_conv_pad False
5 stem_conv False
6 stem_bn False
7 stem_activation False
8 block1a_dwconv False
9 block1a_bn False
10 block1a_activation False
11 block1a_se_squeeze False
12 block1a_se_reshape False
13 block1a_se_reduce False
14 block1a_se_expand False
15 block1a_se_excite False
16 block1a_project_conv False
17 block1a_project_bn False
18 block2a_expand_conv False
19 block2a_expand_bn False
20 block2a_expand_activation False
21 block2a_dwconv_pad False
```

```
210 block6d_dwconv False
211 block6d_bn False
212 block6d_activation False
213 block6d_se_squeeze False
214 block6d_se_reshape False
215 block6d_se_reduce False
216 block6d_se_expand False
217 block6d_se_excite False
218 block6d_project_conv False
219 block6d_project_bn False
220 block6d_drop False
221 block6d_add False
222 block7a_expand_conv False
223 block7a_expand_bn False
224 block7a_expand_activation False
225 block7a_dwconv False
226 block7a_bn False
227 block7a_activation False
228 block7a_se_squeeze False
229 block7a_se_reshape False
230 block7a_se_reduce False
231 block7a_se_expand False
232 block7a_se_excite False
233 block7a_project_conv False
234 block7a_project_bn False
235 top_conv False
236 top_bn False
237 top_activation False
```

Here, all the 237 layers in the base_model are not trainable.

Summarize the model.

```
model_1.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 Input_layer (InputLayer)    [(None, 224, 224, 3)]     0

 Augmented_layer (Sequentia  (None, None, None, 3)     0
 l)

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

 global_average_pooling2d (  (None, 1280)              0
 GlobalAveragePooling2D)

 dense (Dense)               (None, 10)                12810

=================================================================
Total params: 4062381 (15.50 MB)
Trainable params: 12810 (50.04 KB)
Non-trainable params: 4049571 (15.45 MB)
_________________________________________________________________
```

Finetune Model Building

Set the last 10 only for training and set the remaining all as non-trainable.

```
base_model.trainable = True

for layer in base_model.layers[:-10]:
  layer.trainable = False

model_1.compile(loss='categorical_crossentropy',
                optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
                metrics=['accuracy'])
```

```
[17] for layer_number, layer in enumerate(base_model.layers):
       print(layer_number,layer.name,layer.trainable)

0 input_1 False
1 rescaling False
2 normalization False
3 rescaling_1 False
4 stem_conv_pad False
5 stem_conv False
6 stem_bn False
7 stem_activation False
8 block1a_dwconv False
9 block1a_bn False
10 block1a_activation False
11 block1a_se_squeeze False
12 block1a_se_reshape False
13 block1a_se_reduce False
14 block1a_se_expand False
15 block1a_se_excite False
16 block1a_project_conv False
17 block1a_project_bn False
18 block2a_expand_conv False
19 block2a_expand_bn False
20 block2a_expand_activation False
```

```
213 block6d_se_squeeze False
214 block6d_se_reshape False
215 block6d_se_reduce False
216 block6d_se_expand False
217 block6d_se_excite False
218 block6d_project_conv False
219 block6d_project_bn False
220 block6d_drop False
221 block6d_add False
222 block7a_expand_conv False
223 block7a_expand_bn False
224 block7a_expand_activation False
225 block7a_dwconv False
226 block7a_bn False
227 block7a_activation False
228 block7a_se_squeeze True
229 block7a_se_reshape True
230 block7a_se_reduce True
231 block7a_se_expand True
232 block7a_se_excite True
233 block7a_project_conv True
234 block7a_project_bn True
235 top_conv True
236 top_bn True
237 top_activation True
```

Here, except the last 10 layers, remaining first 227 layers in the base_model is non-trainable.

Now train the model again for 25% of the data.

And calculate train_accuracy, train_loss, val_accuracy, and val_loss from the 5th epoch to 10th epoch.

Now fit the model and save it as “history_2”.

```
history_2 = model_1.fit(train_data,
                        epochs=initial_epochs+5,
                        initial_epoch=history_1.epoch[-1],
                        validation_data=test_data,
                        validation_steps=int(0.25*len(test_data)),
                        steps_per_epoch=len(train_data),
                        callbacks=[checkpoint_callback,create_tensorboard_callback(dir_name='tensorflow',experiment_name='Model_1_10_percent_feature_extraction')]
                        )
```

```
Saving TensorBoard log files to: tensorflow/Model_1_10_percent_feature_extraction/20240502-124224
Epoch 5/10
24/24 [==============================] - 155s 6s/step - loss: 0.6509 - accuracy: 0.8120 - val_loss: 0.5159 - val_accuracy: 0.8536
Epoch 6/10
24/24 [==============================] - 104s 4s/step - loss: 0.5235 - accuracy: 0.8333 - val_loss: 0.4553 - val_accuracy: 0.8701
Epoch 7/10
24/24 [==============================] - 143s 6s/step - loss: 0.4443 - accuracy: 0.8773 - val_loss: 0.4517 - val_accuracy: 0.8602
Epoch 8/10
24/24 [==============================] - 144s 6s/step - loss: 0.3995 - accuracy: 0.8707 - val_loss: 0.4405 - val_accuracy: 0.8635
Epoch 9/10
24/24 [==============================] - 138s 6s/step - loss: 0.3627 - accuracy: 0.8893 - val_loss: 0.4310 - val_accuracy: 0.8684
Epoch 10/10
24/24 [==============================] - 146s 6s/step - loss: 0.3231 - accuracy: 0.9093 - val_loss: 0.4545 - val_accuracy: 0.8586
```

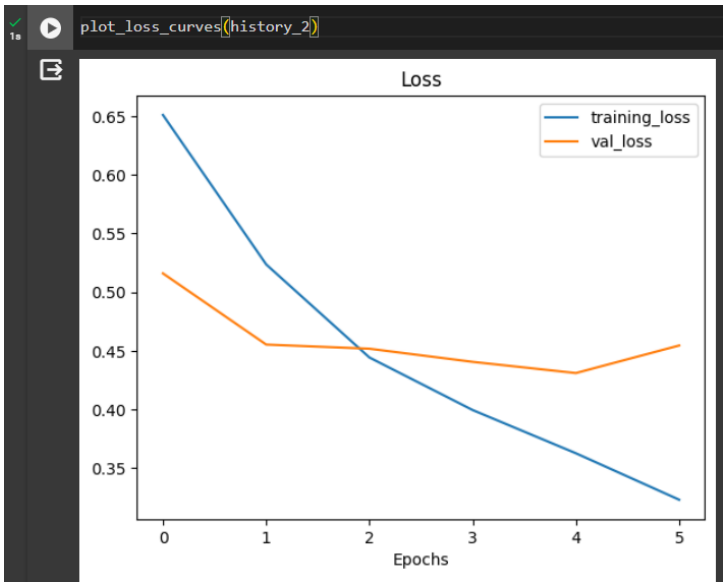
Now train the 100% of data and calculate train_loss, train_accuracy.

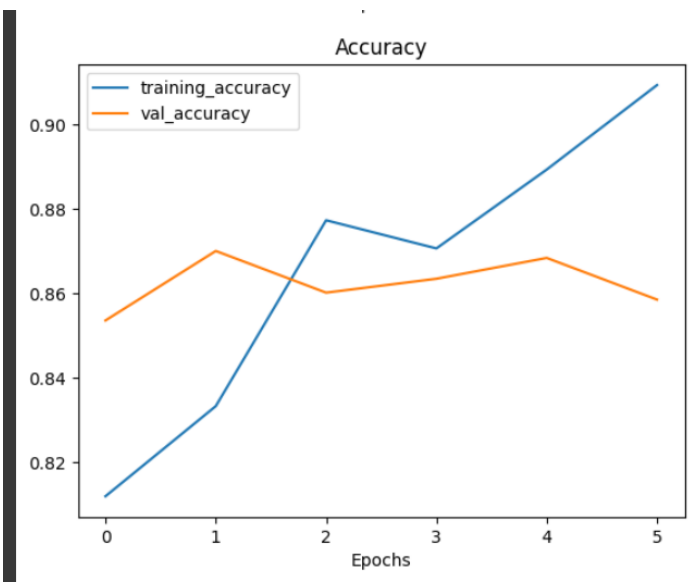
```
[19] fine_tune_results = model_1.evaluate(test_data)

79/79 [==============================] - 179s 2s/step - loss: 0.4244 - accuracy: 0.8616
```

Now plot the loss_curves for the “history_2”.

```
plot_loss_curves(history_2)
```

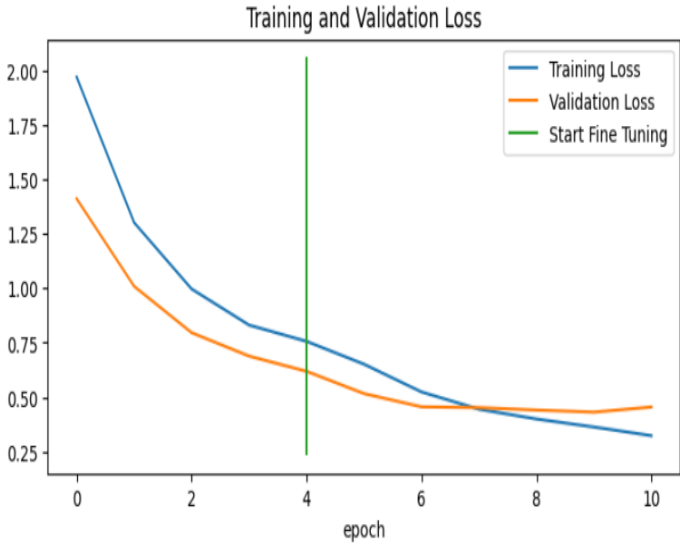




Now, we shall compare the loss_curves for both “history_1” and “history_2”.

```
from helper_functions import compare_historys

compare_historys(original_history=history_1,
                 new_history=history_2)
```



Evaluating the Model

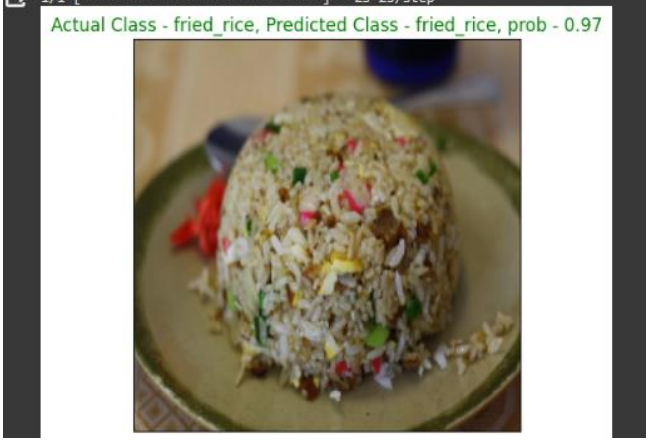
Plot a random single image with prediction

```
import random
import os
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from helper_functions import load_and_prep_image

class_name = random.choice(class_names)
filename = random.choice(os.listdir(test_dir + '/' + class_name))
filepath = test_dir + '/' + class_name + '/' + filename
img = load_and_prep_image(filepath,scale=False)
pred_prob = model_1.predict(tf.expand_dims(img,axis=0))
pred_class = class_names[pred_prob.argmax()]
plt.figure()
fig = plt.imshow(img/255.)
if pred_class==class_name:
  color='g'
else:
  color='r'
plt.title(f"Actual Class - {class_name}, Predicted Class - {pred_class}, prob - {pred_prob.max():.2f}",c=color)
fig.axes.get_xaxis().set_visible(False)
fig.axes.get_yaxis().set_visible(False)
```

```
  color='r'
plt.title(f"Actual Class - {class_name}, Predicted Class - {pred_class}, pr
fig.axes.get_xaxis().set_visible(False)
fig.axes.get_yaxis().set_visible(False)
```

```
1/1 [==============================] - 2s 2s/step
```



Similarly, plot 9 random images with prediction.

```
import matplotlib.pyplot as plt
plt.figure(figsize=(17, 10))
for i in range(9):
  class_name = random.choice(class_names)
  filename = random.choice(os.listdir(test_dir + '/' + class_name))
  filepath = test_dir + '/' + class_name + '/' + filename

  img = load_and_prep_image(filepath, scale=False)
  pred_prob = model_1.predict(tf.expand_dims(img, axis=0))
  pred_class = class_names[pred_prob.argmax()]

  plt.subplot(3, 3, i+1)
  fig = plt.imshow(img/255.)
  if class_name == pred_class:
    title_color = "g"
  else:
    title_color = "r"
  plt.title(f"Actual:{class_name}, Prediction:{pred_class}, prob : {pred_prob.max():.2f}", c=title_color)
  fig.axes.get_xaxis().set_visible(False)
  fig.axes.get_yaxis().set_visible(False)
```

```
1/1 [==============================] - 0s 82ms/step
1/1 [==============================] - 0s 91ms/step
1/1 [==============================] - 0s 82ms/step
1/1 [==============================] - 0s 90ms/step
1/1 [==============================] - 0s 81ms/step
1/1 [==============================] - 0s 87ms/step
1/1 [==============================] - 0s 80ms/step
1/1 [==============================] - 0s 95ms/step
1/1 [==============================] - 0s 82ms/step
```

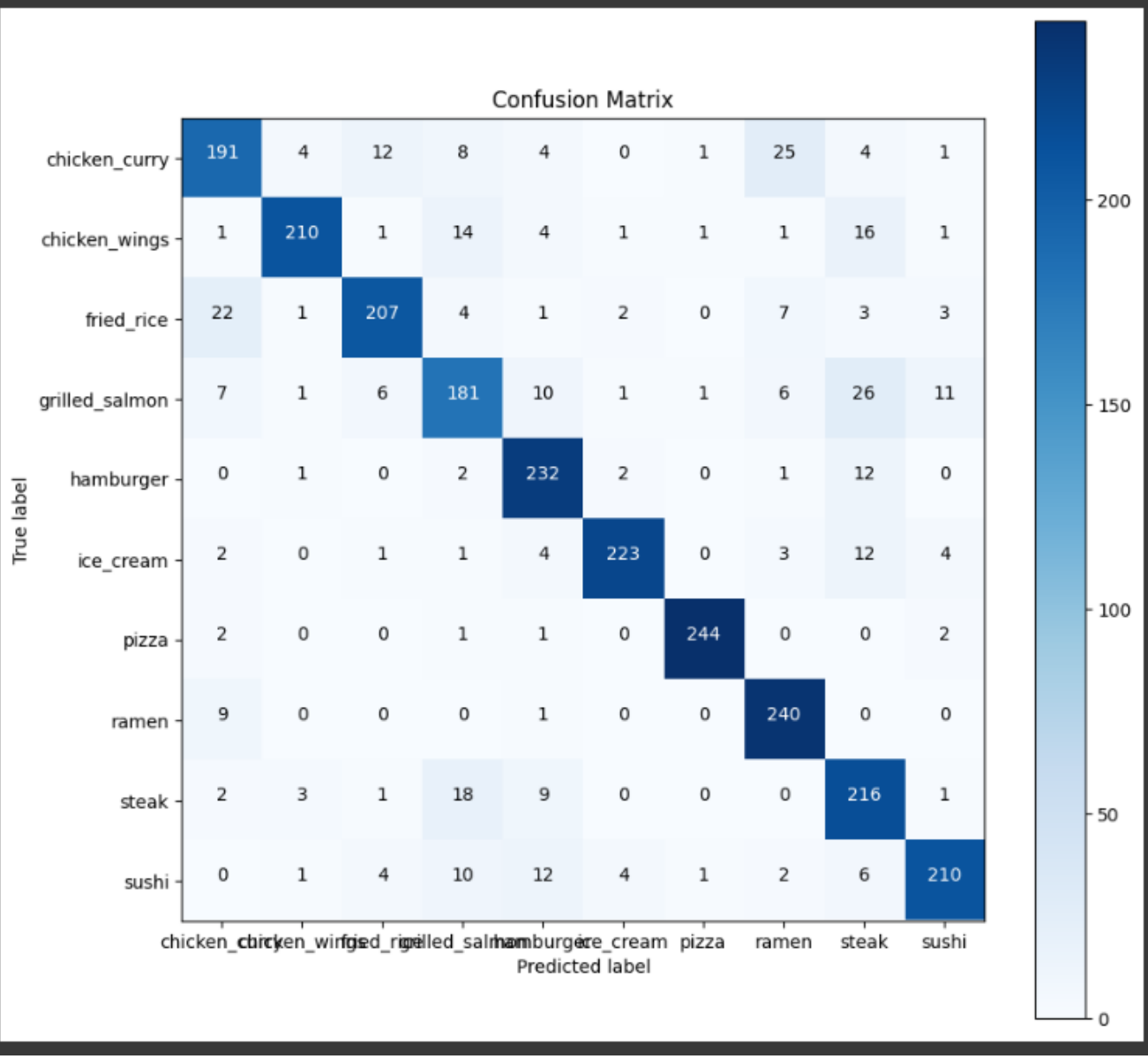
Here out of 9 times, 8 times the model is correct.

Confusion Matrix

Now we shall plot the Confusion Matrix for the 10 food classes.

```
from helper_functions import make_confusion_matrix

make_confusion_matrix(y_true=ylabels,y_pred=pred_classes,classes=class_names,text_size=10)
```



Implementation of Results

```
[35] img=load_and_prep_image("/content/ramen.jpeg",scale=False)
     pred_prob = model_1.predict(tf.expand_dims(img, axis=0))
     pred_class = class_names[pred_prob.argmax()]
     fig = plt.imshow(img/255.)
     plt.title(f"Prediction:{pred_class}, prob : {pred_prob.max():.2f}")
     fig.axes.get_xaxis().set_visible(False)
     fig.axes.get_yaxis().set_visible(False)

1/1 [==============================] - 0s 78ms/step
```

Here, I have taken a random ramen photo from Internet. And here, my model has predicted it correctly as “ramen” with a good probability of “0.97”.

Saving our Model

```
[36] model_1.save("Food_Vision.h5")

/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model as an HDF5 file via `model.save()`. This file format
  saving_api.save_model(
```

Here, we have saved our Model.

Conclusion

In this project, a novel framework for automated food item recognition using machine vision and deep learning techniques has been proposed and implemented. The system addresses the fundamental challenges in food item recognition, including variability in food presentation, occlusions, and scale, by leveraging advanced neural network architectures and feature extraction methods.

Central to the proposed system is the utilization of deep convolutional neural networks (CNNs) for spatial feature extraction and representation learning from food images. Through extensive experimentation and evaluation on diverse food datasets, the system has demonstrated robust performance in accurately identifying and categorizing food items across various cuisines and dining contexts.

The system's effectiveness has been validated through comparative analysis against existing food recognition methods, showcasing its superiority in terms of accuracy and robustness. Moreover, the system's potential applications extend to dietary monitoring apps, restaurant menu analysis tools, and personalized nutrition recommendations, contributing to advancements in health and wellness technologies.

Future research endeavors may explore avenues for improving the system's scalability, computational efficiency, and real-time performance, particularly in the context of mobile and embedded applications. Additionally, there is scope for investigating multi-modal fusion strategies and domain adaptation techniques to enhance the system's adaptability to diverse culinary traditions and environmental conditions.

In conclusion, this project represents a significant contribution to the field of computer vision and has the potential to revolutionize the way we interact with food-related technologies, ultimately promoting healthier eating habits and enhancing quality of life.