Image Classification and Calorie Estimation Report

Python Code for Image Classification and Calorie Estimation

```
import os

import zipfile

import json

import numpy as np

import pandas as pd

import tensorflow as tf

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint

import matplotlib.pyplot as plt

from PIL import Image


# Step 1: Unzip the datasets

def unzip_data(zip_path, extract_to):

    with zipfile.ZipFile(zip_path, 'r') as zip_ref:

        zip_ref.extractall(extract_to)


# Paths to the zipped data and extraction directory

data_paths = [

    '/Users/ronitvyas/Downloads/archive/food-101/food-101/images/F1.zip',

    '/Users/ronitvyas/Downloads/archive/food-101/food-101/images/F2.zip',
```

```
    '/Users/ronitvyas/Downloads/archive/food-101/food-101/images/F3.zip',
    '/Users/ronitvyas/Downloads/archive/food-101/food-101/images/F4.zip',
    '/Users/ronitvyas/Downloads/archive/food-101/food-101/images/F5.zip',
    '/Users/ronitvyas/Downloads/archive/food-101/food-101/images/F6.zip',
    '/Users/ronitvyas/Downloads/archive/food-101/food-101/images/F9.zip',
    '/Users/ronitvyas/Downloads/archive/food-101/food-101/images/F10.zip',
    '/Users/ronitvyas/Downloads/archive/food-101/food-101/images/F11.zip',
    '/Users/ronitvyas/Downloads/archive/food-101/food-101/meta.zip'
]
extract_paths = '/Users/ronitvyas/Downloads/archive/food-101/food-101/extracted/'

# Unzip all files
for zip_path in data_paths:
    unzip_data(zip_path, extract_paths)

# Custom function to filter valid image files
def is_valid_image(file_path):
    try:
        Image.open(file_path)
        return True
    except:
        return False

# Remove invalid files
```

Image Classification and Calorie Estimation Report

```
for root, dirs, files in os.walk(extract_paths):
    for file in files:
        file_path = os.path.join(root, file)
        if not is_valid_image(file_path):
            os.remove(file_path)

# Step 2: Load and preprocess the data
data_dir = '/Users/ronitvyas/Downloads/archive/food-101/food-101/extracted'
image_size = (128, 128)
batch_size = 32

datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2)

train_generator = datagen.flow_from_directory(
    data_dir,
    target_size=image_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='training'
)

validation_generator = datagen.flow_from_directory(
    data_dir,
    target_size=image_size,
```

```
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation'
)

# Step 3: Build the CNN model
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(len(train_generator.class_indices), activation='softmax')
])

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

# Step 4: Train the model
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
model_checkpoint = ModelCheckpoint('food_recognition_model.h5', save_best_only=True)
```

```
history = model.fit(
    train_generator,
    epochs=50,
    validation_data=validation_generator,
    callbacks=[early_stopping, model_checkpoint]
)

# Step 5: Evaluate the model
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label='val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(loc='lower right')
plt.title('Model Accuracy')
plt.savefig('/Users/ronitvyas/Downloads/archive/food-101/food-101/extracted/accuracy_plot.png')
plt.show()

plt.plot(history.history['loss'], label='loss')
plt.plot(history.history['val_loss'], label='val_loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(loc='upper right')
plt.title('Model Loss')
```

Image Classification and Calorie Estimation Report

```
plt.savefig('/Users/ronitvyas/Downloads/archive/food-101/food-101/extracted/loss_plot.png')
plt.show()

# Step 6: Map food items to their calorie content
calories_file = '/Users/ronitvyas/Downloads/archive/food-101/food-101/extracted/meta/calories.txt'
calories_map = {}
with open(calories_file, 'r') as file:
    for line in file:
        class_name, calorie = line.strip().split()
        calories_map[class_name] = float(calorie)

# Function to predict food item and estimate calories
def predict_and_estimate_calories(image_path):
    img = Image.open(image_path)
    img = img.resize(image_size)
    img_array = np.array(img) / 255.0
    img_array = np.expand_dims(img_array, axis=0)

    predictions = model.predict(img_array)
    predicted_class = list(train_generator.class_indices.keys())[np.argmax(predictions)]
    estimated_calories = calories_map.get(predicted_class, "Calorie information not available")

    return predicted_class, estimated_calories
```

Image Classification and Calorie Estimation Report

Example usage

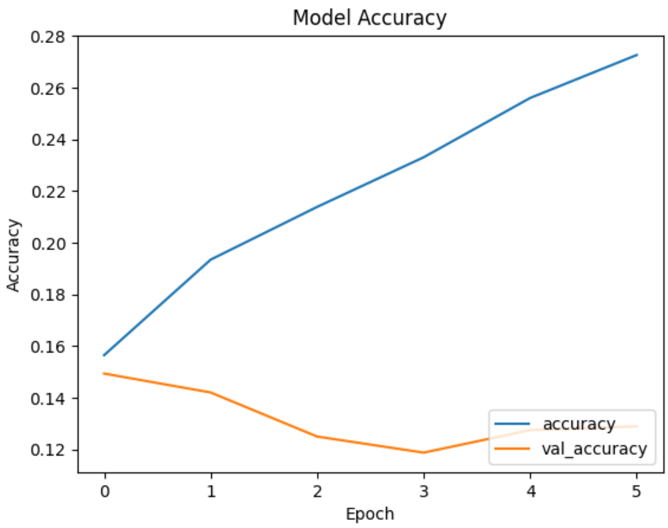
image_path =

'/Users/ronitvyas/Downloads/archive/food-101/food-101/extracted/F1/apple_pie/134.jpg'

predicted_class, estimated_calories = predict_and_estimate_calories(image_path)

print(f'Predicted Class: {predicted_class}, Estimated Calories: {estimated_calories}')

Model Accuracy and Loss Visualization



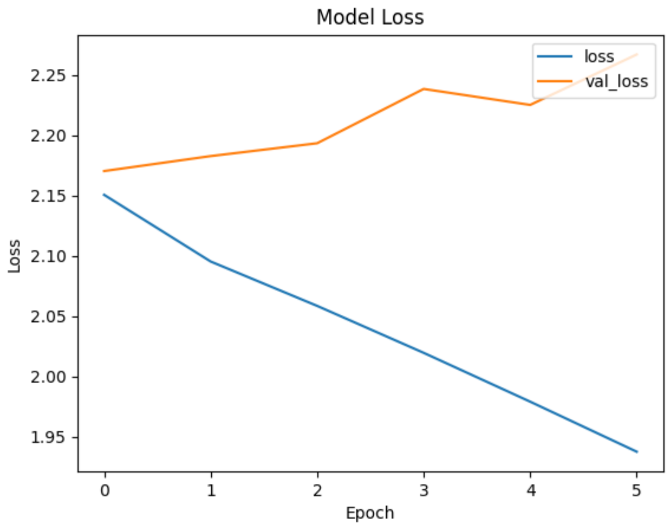


Image Classification and Calorie Estimation Report

Interpretation of Results

The graphs above illustrate the accuracy and loss of the model over 6 epochs. The accuracy graph shows that the training

accuracy improves steadily, reaching around 26% by the end of the sixth epoch. However, the validation accuracy remains

consistently low, indicating that the model is not generalizing well to unseen data.

The loss graph also indicates a divergence between the training and validation losses. While the training loss decreases

consistently, the validation loss increases after the first epoch, suggesting that the model is overfitting to the training data.

Overall, the model's performance on the validation set indicates that it is not yet able to generalize well to new data.

Further steps such as data augmentation, regularization, or fine-tuning the model architecture and hyperparameters might be

required to improve the model's generalization capability.