Food Recognition for Calorie Estimation

Course: Applied Machine Learning

Supervisor: Aiymbay Sunggat Group: BDA-2205

Team members: Alyona Abaidulina, Ernazar Ermuratov

Date: 19.11.2024

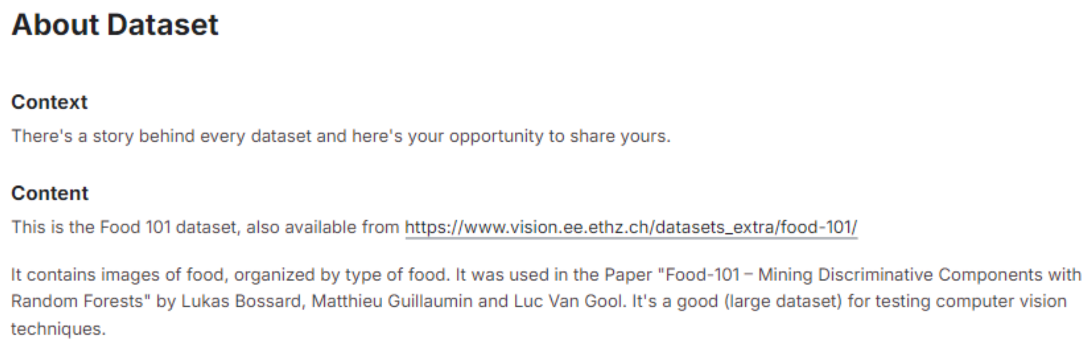
Table of contents

1. Project Overview

This project aims to build a deep learning model for classifying food from image and predicting its calorie content. The project was done in two parts, the initial model is using a pre-trained EfficientNet-B0 model, then we worked on improving it and built an enhanced FoodCalorieClassifier using better metrics and advanced layer architecture. In training, accuracy was taken as the main metric, but precision, recall and F1 score were also taken into account to provide the most unbiased evaluation.

2. Dataset and Preprocessing

This project is based on a dataset from the open source Kaggle database. The Food 101 dataset will contain 101 food categories with 1000 photos per class.



About Dataset

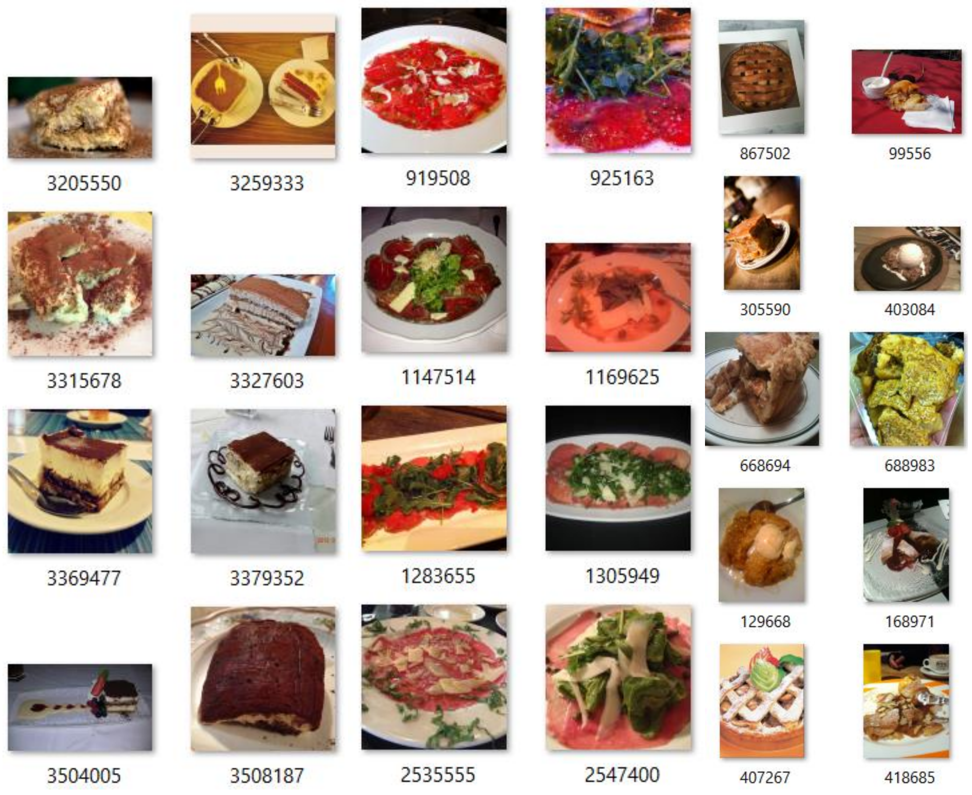
Context

There's a story behind every dataset and here's your opportunity to share yours.

Content

This is the Food 101 dataset, also available from https://www.vision.ee.ethz.ch/datasets_extra/food-101/

It contains images of food, organized by type of food. It was used in the Paper "Food-101 – Mining Discriminative Components with Random Forests" by Lukas Bossard, Matthieu Guillaumin and Luc Van Gool. It's a good (large dataset) for testing computer vision techniques.



Link to dataset: https://www.kaggle.com/datasets/dansbecker/food-101

Data preparation

For our task, we selected 20 classes of dishes in order to reduce the computational resources of training the model.

In the end, our dataset contained the following classes:

1. apple_pie
2. beef_tartare
3. breakfast_burrito
4. carrot_cake
5. sushi
6. baby_back_ribs
7. beignets
8. caesar_salad
9. spaghetti_bolognese
10. tacos
11. baklava
12. bibimbap
13. cannoli
14. steak
15. tiramisu
16. beef_carpaccio
17. bread_pudding
18. caprese_salad
19. strawberry_shortcake
20. waffles

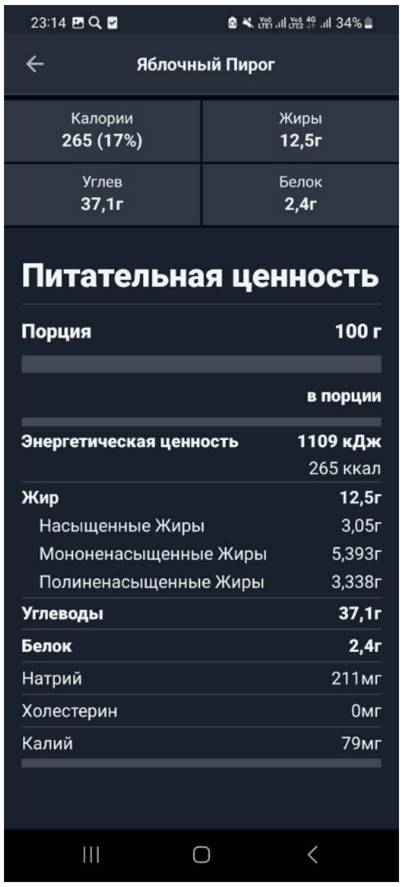
All images have been scaled to 224 by 224 pixels

Calories labeling

To determine the caloric content of the dishes, we used the FatSecret nutritional app, which provides data on composition and energy value for a large number of dishes.

In the end, we obtained the following correspondence of classes to their caloric value per 100 grams of food:

calories_per_100g = { "apple_pie": 265, "baby_back_ribs": 381, "baklava": 428, "beef_tartare": 157, "bread_pudding": 350, "caesar_salad": 188, "caprese_salad": 142, "strawberry_shortcake": 274, "cannoli": 254, "waffles": 310, "breakfast_burrito": 225, "sushi": 157, "beef_carpaccio": 168, "spaghetti_bolognese": 260, "tiramisu": 283, "carrot_cake": 310, "steak": 230, "beignets": 399, "bibimbap": 146, "tacos": 216
}


23:14 34%

Яблочный Пирог

Калории 265 (17%)	Жиры 12,5г
Углев 37,1г	Белок 2,4г

Питательная ценность

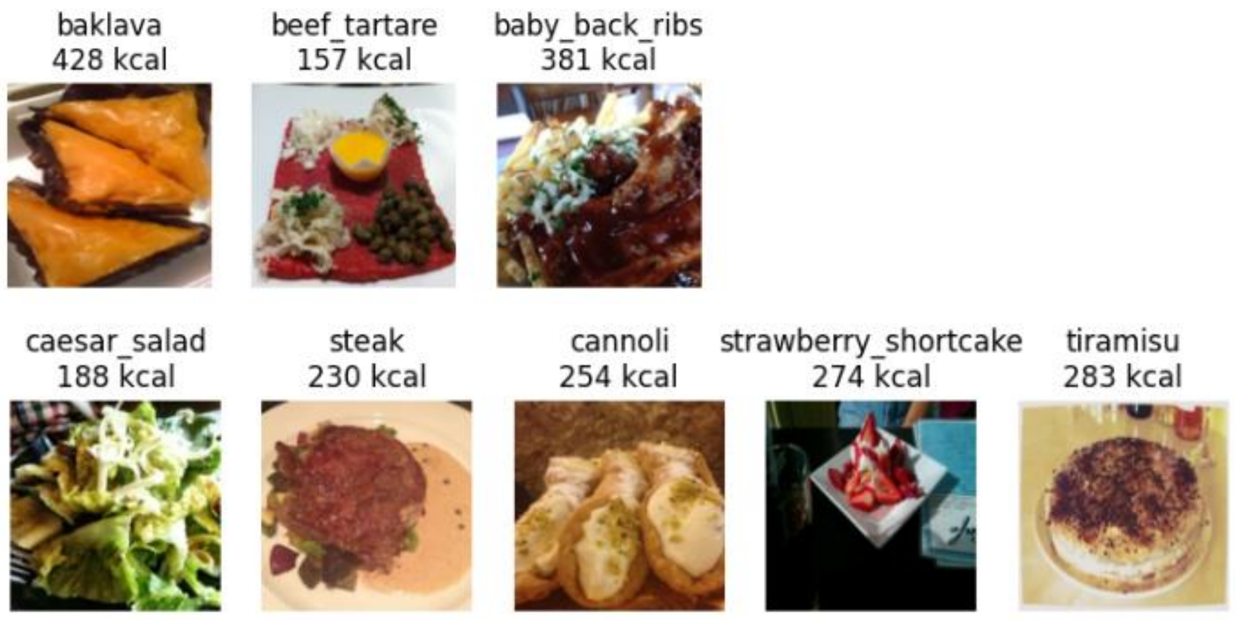
Порция	**100 г**
	в порции
Энергетическая ценность	**1109 кДж** 265 ккал
Жир	**12,5г**
Насыщенные Жиры	3,05г
Мононенасыщенные Жиры	5,393г
Полиненасыщенные Жиры	3,338г
Углеводы	**37,1г**
Белок	**2,4г**
Натрий	211мг
Холестерин	0мг
Калий	79мг

Enhanced Calories Labeling:
This approach dynamically estimated calorie content based on portion size by calculating food volume through image area and predefined depth values. Volume was converted to weight using food-specific densities, and calories were derived from the weight using predefined calories-per-gram values. While this method provided more realistic calorie estimates, its reliance on computationally intensive volume and depth estimation made it resource-heavy and less scalable for large datasets. We just transformed train and test sets to this labeling but due to not have enough resources we decided to adjust our preprocessing step and remain this part for future development.

Data Splitting:
The dataset was split into three parts:

- **Training Set:** 320 images x20 classes
- **Validation Set:** 80 images x20 classes
- **Test Set:** 200images x20 classes

Final dataset samples:



3. Baseline model

The architecture consists of two main components: an EfficientNet-B0-based feature extraction network and two branching prediction heads for classification and calorie estimation.

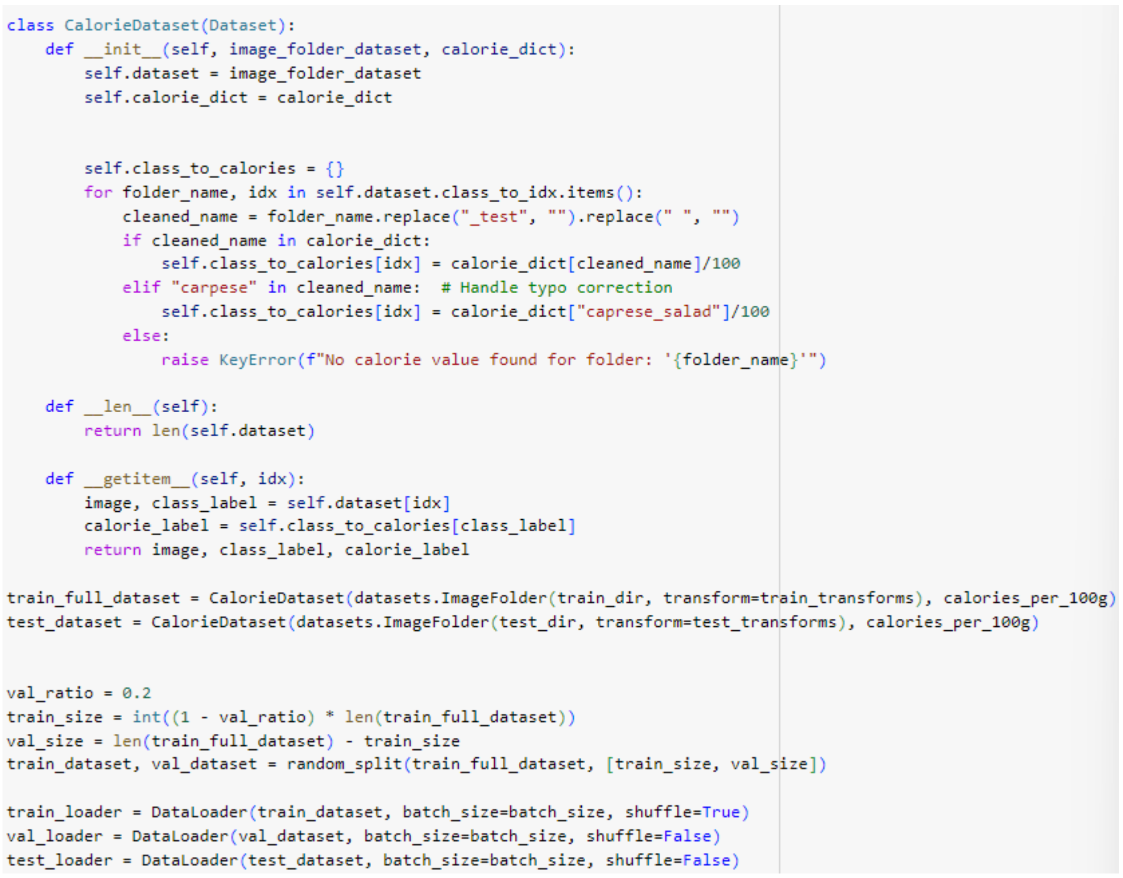
Model Architecture:

EfficientNet-B0 Backbone: In the proposed model, EfficientNet-B0 is adopted as the feature extraction layer. We use the EfficientNet-B0 model pre-trained on ImageNet for feature extraction since using pre-trained models help the model learn features from images before fine-tuning them on the food dataset.
The backbone's classifier layer is removed, and the output from the feature extractor is passed through two separate fully connected (linear) layers:

1) Food Classification Head: This head provides the class label for each image that is given to this classifier (as apple pie and etc.)

2) Calorie Prediction Head: This head means to estimate the calorie content of food picture and yields real value which denotes calorie per 100g of food item.

Data Loading and preprocessing:
A custom CalorieDataset class is designed to load the image data, and the corresponding calorie values.
Depending on the nature of the images, they are transformed by fixing their sizes and normalizing the pixel values so that the images may fit the requirements for feed into a model.



```python
class CalorieDataset(Dataset):
    def __init__(self, image_folder_dataset, calorie_dict):
        self.dataset = image_folder_dataset
        self.calorie_dict = calorie_dict


        self.class_to_calories = {}
        for folder_name, idx in self.dataset.class_to_idx.items():
            cleaned_name = folder_name.replace("_test", "").replace(" ", "")
            if cleaned_name in calorie_dict:
                self.class_to_calories[idx] = calorie_dict[cleaned_name]/100
            elif "carpese" in cleaned_name:  # Handle typo correction
                self.class_to_calories[idx] = calorie_dict["caprese_salad"]/100
            else:
                raise KeyError(f"No calorie value found for folder: '{folder_name}'")

    def __len__(self):
        return len(self.dataset)

    def __getitem__(self, idx):
        image, class_label = self.dataset[idx]
        calorie_label = self.class_to_calories[class_label]
        return image, class_label, calorie_label

train_full_dataset = CalorieDataset(datasets.ImageFolder(train_dir, transform=train_transforms), calories_per_100g)
test_dataset = CalorieDataset(datasets.ImageFolder(test_dir, transform=test_transforms), calories_per_100g)


val_ratio = 0.2
train_size = int((1 - val_ratio) * len(train_full_dataset))
val_size = len(train_full_dataset) - train_size
train_dataset, val_dataset = random_split(train_full_dataset, [train_size, val_size])

train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
val_loader = DataLoader(val_dataset, batch_size=batch_size, shuffle=False)
test_loader = DataLoader(test_dataset, batch_size=batch_size, shuffle=False)
```

Data augmentation:

For the first model, we applied the most minimal augmentation consisting of

- scaling the image to 224 pixels
- normalising the pixel values of the three colour channels and
- normalising the calorie values.

Parameters:

For the first model, we chose

- batch_size = 64
- num_epochs = 10
- learning_rate = 0.00001

Loss Functions and Optimization:

The model uses two loss functions:

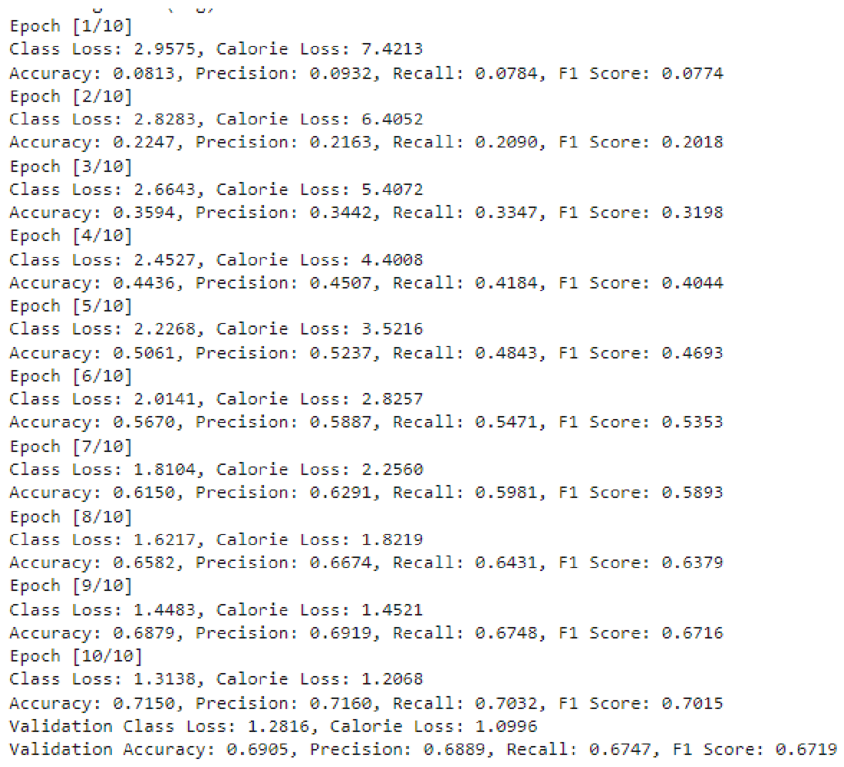
1) Classification Loss (CrossEntropyLoss): Evaluates how well the model classifies food images into categories.
2) Calorie Loss (MSE Loss): Measures the difference between predicted and actual calorie values.

The total loss is a weighted sum of both losses, with the calorie loss weighted by 0.01 to prevent it from overpowering the classification task. This ensures the model focuses more on food classification while still learning to predict calorie content.

The optimizer used is Adam, known for its adaptive learning rate and efficiency in training deep learning models.

Training Procedure:
The model is trained for 10 epochs, following these steps:
Forward Pass: Images are passed through the network to generate class and calorie predictions.
Loss Calculation: Both classification and calorie losses are computed, and the total loss is determined.
Backpropagation and Optimization: The optimizer adjusts the model’s weights based on the total loss.
Metrics Tracking:
Key performance metrics are calculated, including:

- Accuracy: The percentage of correctly classified food images.
- Precision: The ability to correctly classify images without false positives.
- Recall: The ability to correctly identify all relevant food items.
- F1 Score: A balanced metric combining precision and recall.

Results:

```
Epoch [1/10]
Class Loss: 2.9575, Calorie Loss: 7.4213
Accuracy: 0.0813, Precision: 0.0932, Recall: 0.0784, F1 Score: 0.0774
Epoch [2/10]
Class Loss: 2.8283, Calorie Loss: 6.4052
Accuracy: 0.2247, Precision: 0.2163, Recall: 0.2090, F1 Score: 0.2018
Epoch [3/10]
Class Loss: 2.6643, Calorie Loss: 5.4072
Accuracy: 0.3594, Precision: 0.3442, Recall: 0.3347, F1 Score: 0.3198
Epoch [4/10]
Class Loss: 2.4527, Calorie Loss: 4.4008
Accuracy: 0.4436, Precision: 0.4507, Recall: 0.4184, F1 Score: 0.4044
Epoch [5/10]
Class Loss: 2.2268, Calorie Loss: 3.5216
Accuracy: 0.5061, Precision: 0.5237, Recall: 0.4843, F1 Score: 0.4693
Epoch [6/10]
Class Loss: 2.0141, Calorie Loss: 2.8257
Accuracy: 0.5670, Precision: 0.5887, Recall: 0.5471, F1 Score: 0.5353
Epoch [7/10]
Class Loss: 1.8104, Calorie Loss: 2.2560
Accuracy: 0.6150, Precision: 0.6291, Recall: 0.5981, F1 Score: 0.5893
Epoch [8/10]
Class Loss: 1.6217, Calorie Loss: 1.8219
Accuracy: 0.6582, Precision: 0.6674, Recall: 0.6431, F1 Score: 0.6379
Epoch [9/10]
Class Loss: 1.4483, Calorie Loss: 1.4521
Accuracy: 0.6879, Precision: 0.6919, Recall: 0.6748, F1 Score: 0.6716
Epoch [10/10]
Class Loss: 1.3138, Calorie Loss: 1.2068
Accuracy: 0.7150, Precision: 0.7160, Recall: 0.7032, F1 Score: 0.7015
Validation Class Loss: 1.2816, Calorie Loss: 1.0996
Validation Accuracy: 0.6905, Precision: 0.6889, Recall: 0.6747, F1 Score: 0.6719
```
Total accuracy on test dataset = 69%

4. Enhanced data augmentation and parameter adjustments

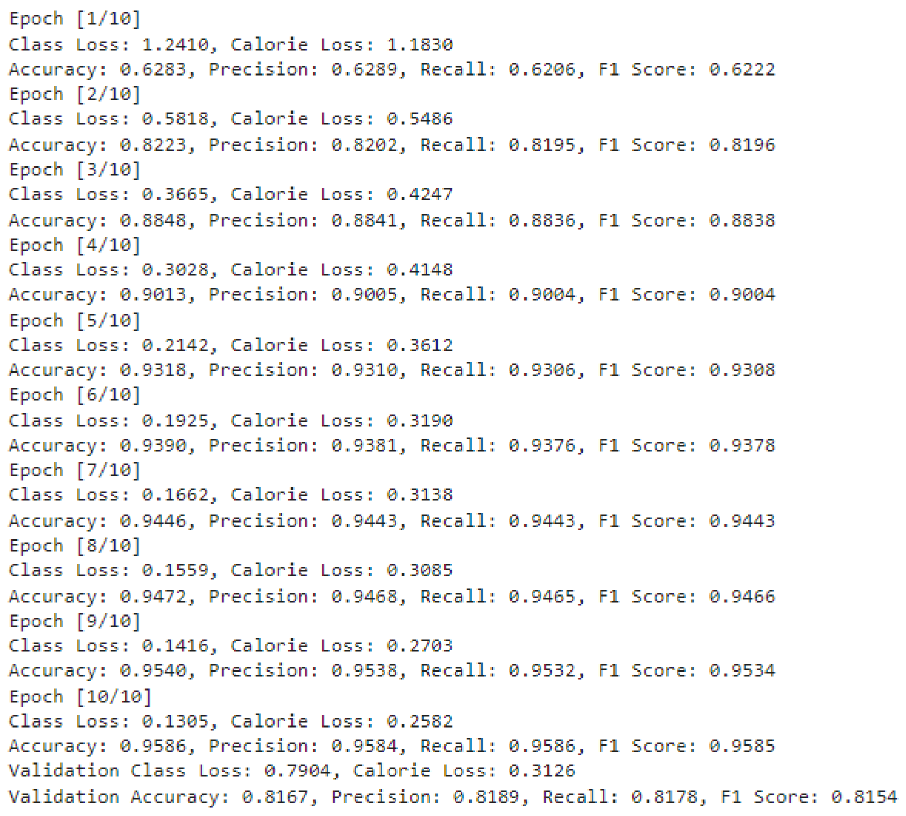
In this phase, we made several changes to the data augmentation pipeline and model training parameters, resulting in a significant improvement of +12% in classification accuracy.
Augmentation Changes:
For the training dataset, we applied a combination of transformations:

- RandomResizedCrop (224, scale=(0.8, 1.0)): This helps the model become more robust to scale variations by cropping and resizing images within a defined range.
- RandomHorizontalFlip: Introduces variability by randomly flipping the images horizontally, which helps the model generalize better.
- RandomRotation (10): Allows the model to learn rotational invariance by rotating images randomly within a ±10-degree range.

Training Parameters:

- We increased the batch size to 32, allowing the model to process more data at once and making training more stable.
- We set the learning rate to 0.0005, which balances model convergence speed and stability, ensuring the model doesn't overshoot optimal weights during training.

These changes in data augmentation and training parameters enabled the model to better generalize across varied images, leading to improved accuracy and better overall performance on the task.



```
Epoch [1/10]
Class Loss: 1.2410, Calorie Loss: 1.1830
Accuracy: 0.6283, Precision: 0.6289, Recall: 0.6206, F1 Score: 0.6222
Epoch [2/10]
Class Loss: 0.5818, Calorie Loss: 0.5486
Accuracy: 0.8223, Precision: 0.8202, Recall: 0.8195, F1 Score: 0.8196
Epoch [3/10]
Class Loss: 0.3665, Calorie Loss: 0.4247
Accuracy: 0.8848, Precision: 0.8841, Recall: 0.8836, F1 Score: 0.8838
Epoch [4/10]
Class Loss: 0.3028, Calorie Loss: 0.4148
Accuracy: 0.9013, Precision: 0.9005, Recall: 0.9004, F1 Score: 0.9004
Epoch [5/10]
Class Loss: 0.2142, Calorie Loss: 0.3612
Accuracy: 0.9318, Precision: 0.9310, Recall: 0.9306, F1 Score: 0.9308
Epoch [6/10]
Class Loss: 0.1925, Calorie Loss: 0.3190
Accuracy: 0.9390, Precision: 0.9381, Recall: 0.9376, F1 Score: 0.9378
Epoch [7/10]
Class Loss: 0.1662, Calorie Loss: 0.3138
Accuracy: 0.9446, Precision: 0.9443, Recall: 0.9443, F1 Score: 0.9443
Epoch [8/10]
Class Loss: 0.1559, Calorie Loss: 0.3085
Accuracy: 0.9472, Precision: 0.9468, Recall: 0.9465, F1 Score: 0.9466
Epoch [9/10]
Class Loss: 0.1416, Calorie Loss: 0.2703
Accuracy: 0.9540, Precision: 0.9538, Recall: 0.9532, F1 Score: 0.9534
Epoch [10/10]
Class Loss: 0.1305, Calorie Loss: 0.2582
Accuracy: 0.9586, Precision: 0.9584, Recall: 0.9586, F1 Score: 0.9585
Validation Class Loss: 0.7904, Calorie Loss: 0.3126
Validation Accuracy: 0.8167, Precision: 0.8189, Recall: 0.8178, F1 Score: 0.8154
```

5. Enhanced model: 1) Randomized search for parameters tunning

To further optimize our model, we conducted a randomized search over a predefined set of hyperparameters. The grid of parameters we tested included learning rate, batch size, weight decay, and optimizer type. We ran the search for 10 random combinations of these hyperparameters.

1) **Hyperparameter Grid:**
   - Learning Rate: [1e-3, 1e-4]
   - Batch Size: [16, 32]
   - Weight Decay: [1e-5]
   - Optimizer: ['adam', 'sgd']
2) **Randomized Search Procedure:**

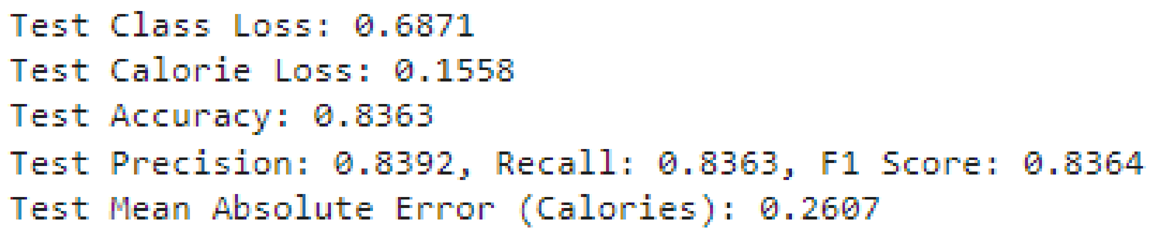
   - For each combination, the model was trained for 5 epochs, and validation accuracy was computed after each training run.
   - We used Adam and SGD optimizers, and tried different batch sizes (16 and 32) to find the best configuration.
3) **Key Findings:**

The best validation accuracy of 0.8482 was achieved with the following parameters:

- Learning Rate: 0.0001
- Batch Size: 32
- Weight Decay: 1e-5
- Optimizer: Adam

This result indicates that a lower learning rate combined with a batch size of 32 and the Adam optimizer provided the best performance. We observed that this configuration significantly outperformed others, with accuracy improvements reaching up to +3% compared to the baseline.

This enhanced model gives us following results:



```
Test Class Loss: 0.6871
Test Calorie Loss: 0.1558
Test Accuracy: 0.8363
Test Precision: 0.8392, Recall: 0.8363, F1 Score: 0.8364
Test Mean Absolute Error (Calories): 0.2607
```

This show pretty good results in classification food as well as prediction of calories.

6. Enhanced model: 2) Architecture refinement

1) Added Layers in the Classifier Block: The classifier block has been expanded with the following layers:

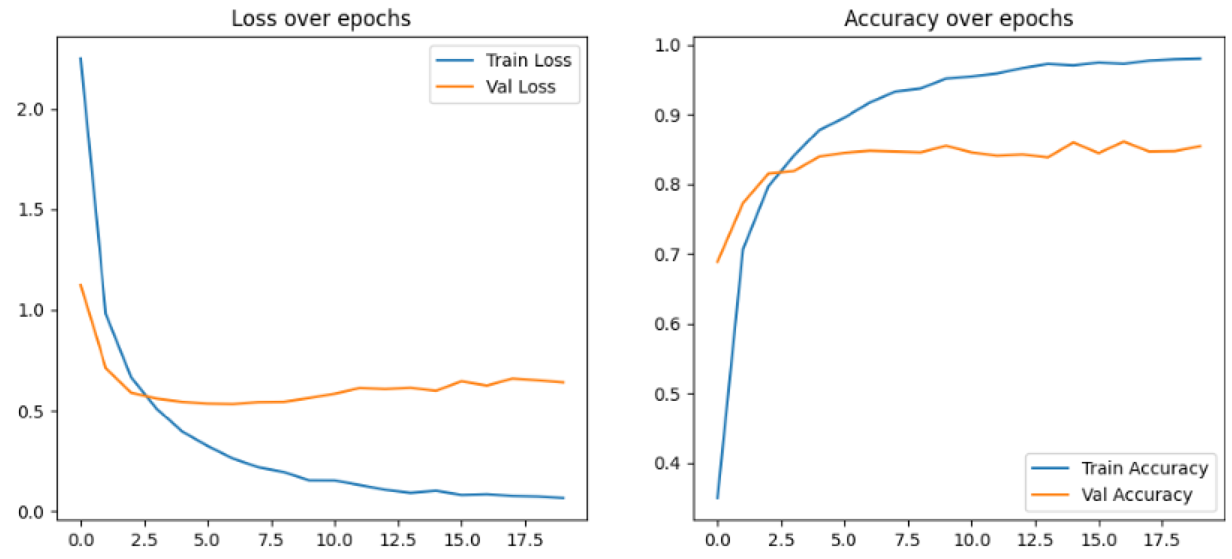
- Fully Connected (Dense) Layer: 512 neurons to enhance the model's ability to capture complex patterns.
- Activation Layer (ReLU): Introduced after the dense layer to add non-linearity. Helps the model learn complex decision boundaries.
- Dropout Layer (Rate = 0.5): Randomly sets 50% of the neurons to zero during training to prevent overfitting. Ensures robust generalization on unseen data.
- Output Layer:A dense layer with neurons equal to the number of dish categories. Softmax activation for generating probability distributions over the classes.

2) Added Layers in the Calorie Predictor Block: Similarly, the regression branch was enhanced with:

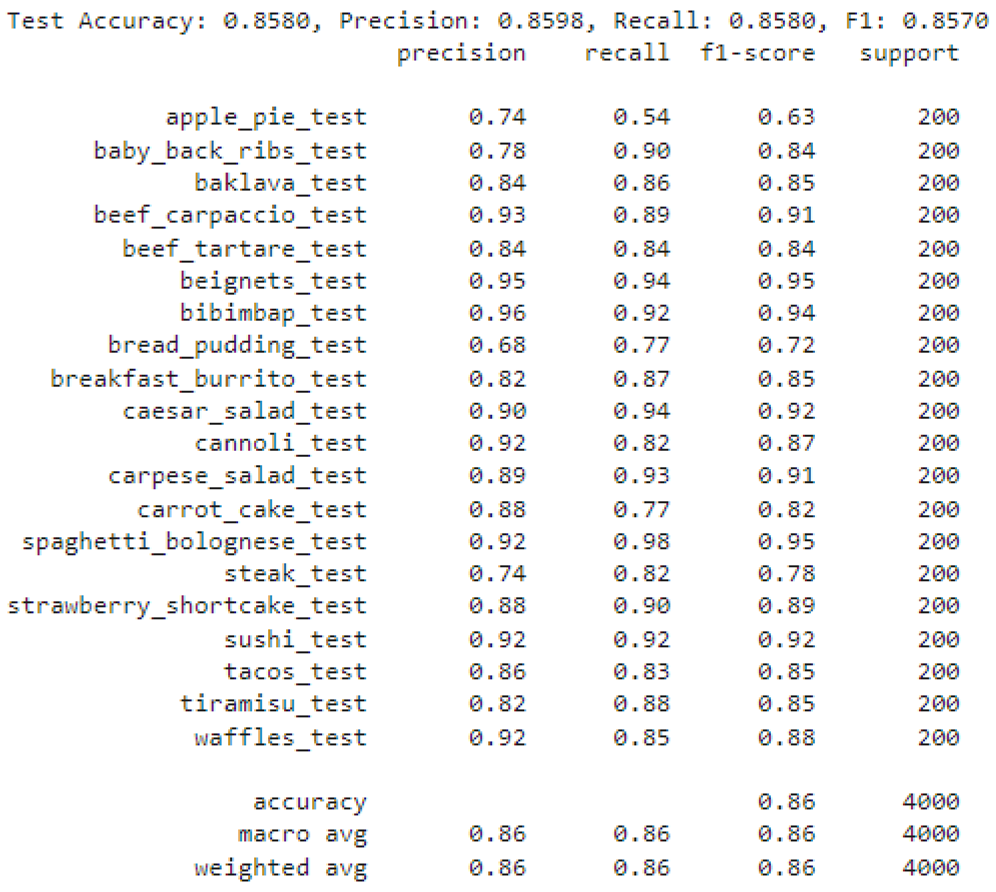
- Fully Connected (Dense) Layer: 512 neurons for learning high-level patterns necessary for calorie prediction.
- Activation Layer (ReLU): Introduced after the dense layer for better representation of non-linear relationships in calorie prediction.
- Dropout Layer (Rate = 0.5): Reduces overfitting, particularly in regression tasks, where the model can easily memorize specific calorie values.
- Output Layer:A dense layer with a single neuron for predicting calorie count. No activation function, as the regression output is continuous.

Impact of the Changes

Improved Classification Accuracy: By **+3%** on the test set due to the deeper and regularized classifier block.



Visualization to understand improvements of Accuracy

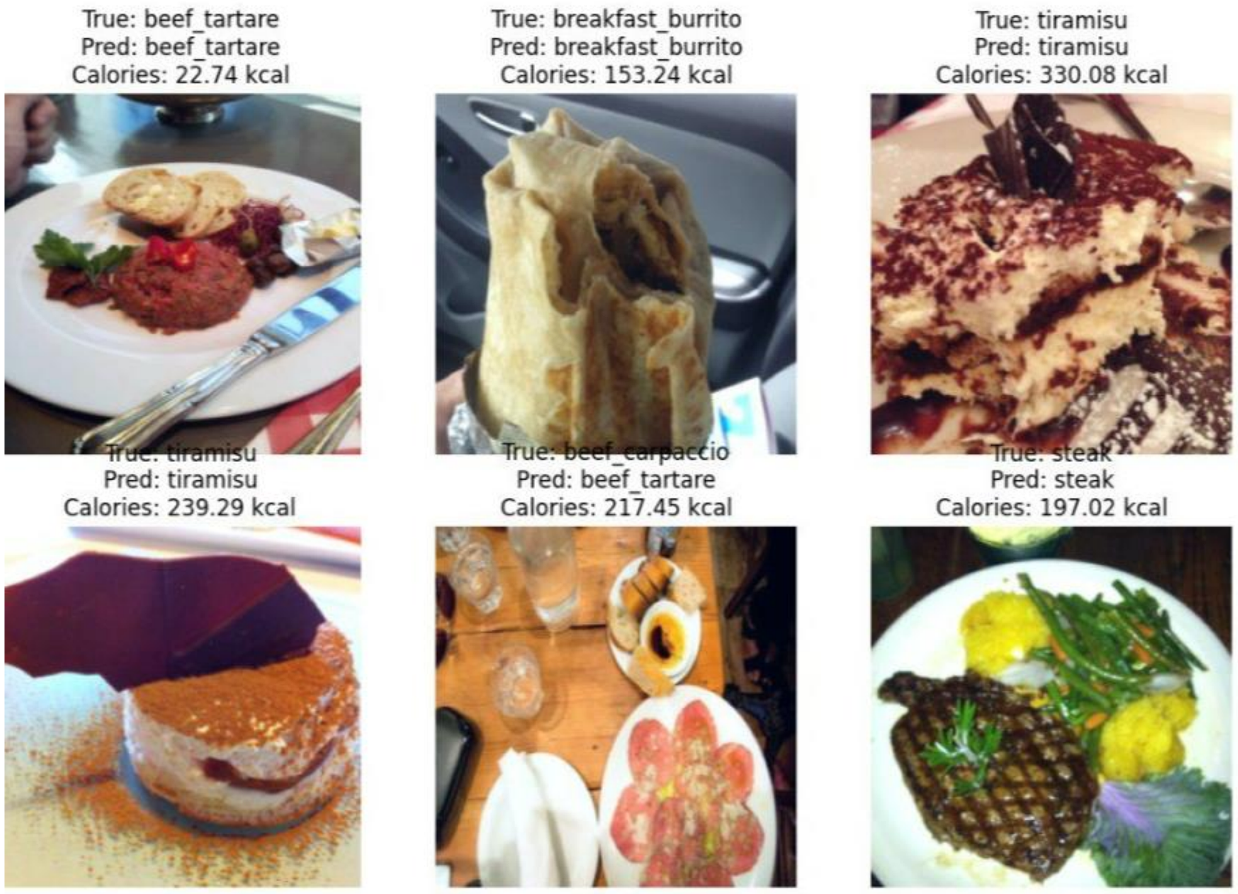


```
Test Accuracy: 0.8580, Precision: 0.8598, Recall: 0.8580, F1: 0.8570
                           precision    recall  f1-score   support

          apple_pie_test       0.74      0.54      0.63       200
      baby_back_ribs_test       0.78      0.90      0.84       200
            baklava_test       0.84      0.86      0.85       200
     beef_carpaccio_test       0.93      0.89      0.91       200
       beef_tartare_test       0.84      0.84      0.84       200
           beignets_test       0.95      0.94      0.95       200
           bibimbap_test       0.96      0.92      0.94       200
      bread_pudding_test       0.68      0.77      0.72       200
  breakfast_burrito_test       0.82      0.87      0.85       200
       caesar_salad_test       0.90      0.94      0.92       200
            cannoli_test       0.92      0.82      0.87       200
      carpese_salad_test       0.89      0.93      0.91       200
        carrot_cake_test       0.88      0.77      0.82       200
 spaghetti_bolognese_test       0.92      0.98      0.95       200
              steak_test       0.74      0.82      0.78       200
strawberry_shortcake_test       0.88      0.90      0.89       200
              sushi_test       0.92      0.92      0.92       200
              tacos_test       0.86      0.83      0.85       200
           tiramisu_test       0.82      0.88      0.85       200
            waffles_test       0.92      0.85      0.88       200

                accuracy                           0.86      4000
               macro avg       0.86      0.86      0.86      4000
            weighted avg       0.86      0.86      0.86      4000
```

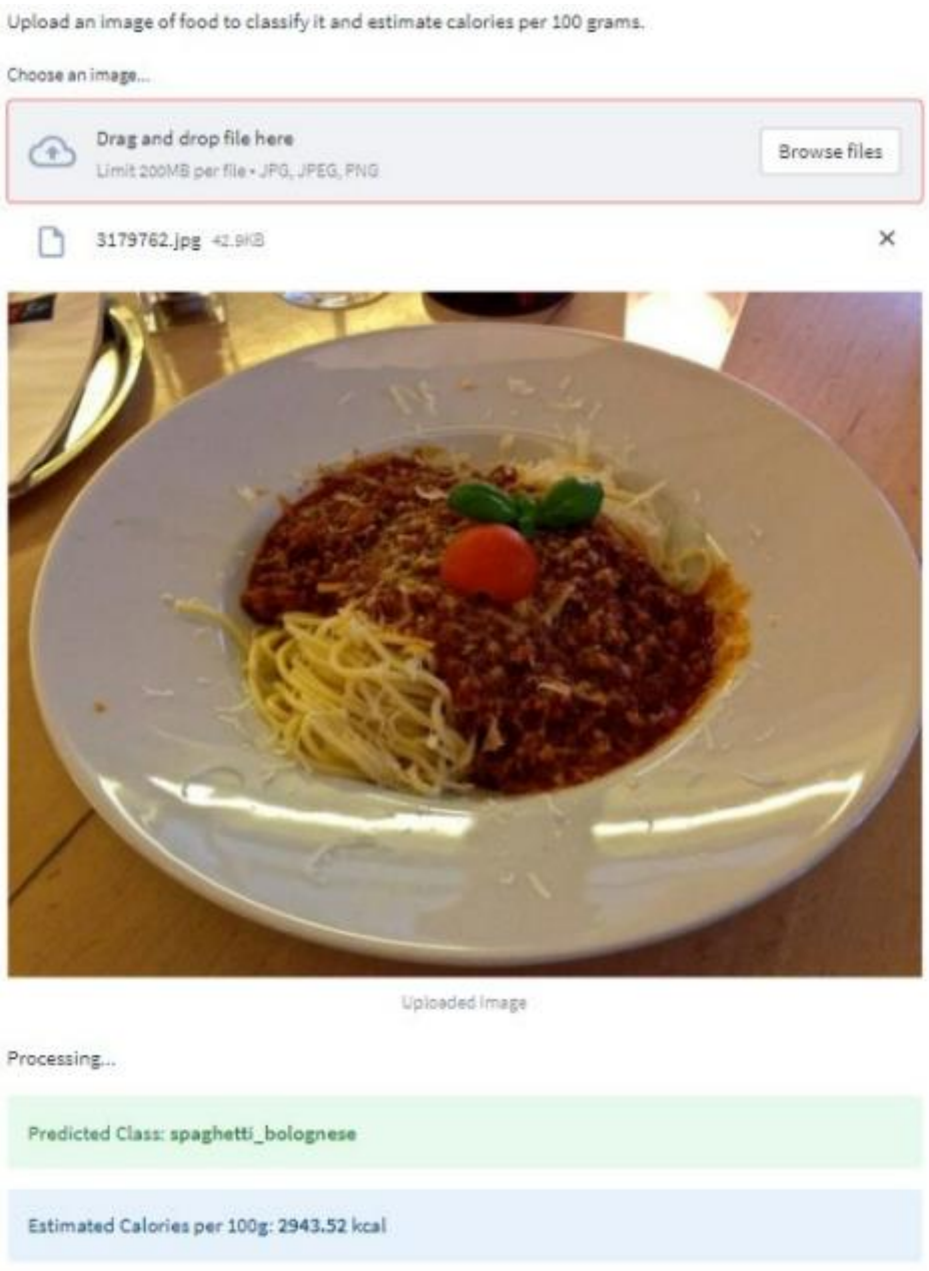
Each metrics overall and on each class



Predicted Images

7. **Streamlit**

A user-friendly web application was developed using Streamlit to facilitate the testing of the proposed model. This application allows users to upload an image, which is then processed by the model to first classify the specific food type and subsequently predict its calorie content. This streamlined interface ensures accessibility while showcasing the practical applicability of the model for real-world scenarios.


Upload an image of food to classify it and estimate calories per 100 grams.

Choose an image...

3179762.jpg 42.9KB

Uploaded Image

Processing...

Predicted Class: spaghetti_bolognese

Estimated Calories per 100g: 2943.52 kcal

8. Conclusion

To sum up, this work effectively proved the effective establishment of deep-learning-based food identification and calorie calculation to enhance human diet is feasible and workable. Thus, applying a step by step procedure, which included the initial baseline model and subsequent improvements, it was possible to ensure higher effectiveness and generalization of the solution.

The key outcomes of the project include:

- Utilisation of pre-trained EfficientNet-B0 for feature extraction helps the model have strong starting performance as the test accuracy recorded a 69%.
- Application of improved data augmentation tools and parameters tuning, which in turn led to the increase of the model accuracy by 12 percent.
- Randomized hyperparameters search improving configuration on validation to new best configuration 83%.

- Relatively, new architectural improvements in both the classification part and the calorie prediction part include the addition more layers that are an indication of the deeper learning from the data and dropout regulization which as resulted to an improved accuracy by +3% .

Link to the GitHub: https://github.com/Yernazzar/foodrecognitiondl.git

Link to the Streamlit: https://jbtghpnczu6fgf43atqufs.streamlit.app/