# Food Identification and Segmentation with Calorie Calculation for Complete Meals IIITD ML CSE343/ECE363

## Abstract

*In today's health-conscious environment, monitoring calorie intake has become essential to maintaining a healthy lifestyle. This project focuses on developing an automated system to accurately identify and segment food items from images and calculate their calorie counts. Using machine learning techniques, specifically Convolutional Neural Networks (CNNs) and decision trees, we aim to make nutritional monitoring more accessible and effective. The project incorporates cutting-edge image processing technologies and models to achieve high accuracy in food recognition,*

## 1. Introduction

The growing demand for personalized health tools has driven significant research into automating dietary monitoring through image-based analysis. One of the most critical components in diet management is accurately measuring calorie intake. Traditional methods rely on manual input, which can be tedious and error-prone. This project aims to develop an efficient system that can automatically identify and calculate the calorie content of food items using machine learning models (MLP and comparing them with Deep Learning Algorithms) with image processing to extract key features of food and predict the calories.

### 1.1. Paper reviews

This review examines two research papers [2]that focus on applying computer vision and machine learning techniques for food recognition and nutrition analysis. Both papers emphasize the increasing role of artificial intelligence (AI) in automating food identification from images and estimating nutritional content. Our analysis found that Convolutional Neural Networks (CNNs) and deep learning models are fundamental to improving dietary tracking, nutrition labeling, and food safety.

The [1] primary approach used in the papers revolves around CNNs, which are highly effective in processing and classifying visual data. In particular, CNNs excel in identifying food types and estimating portion sizes by extracting patterns from images. The first paper proposes a detailed methodology that includes data collection, preprocessing, feature extraction, and model training using the YOLOv2 object detection method. This system is designed to perform real-time food recognition, allowing users to estimate portion sizes and nutritional values directly from images. After classification, the nutritional information is retrieved from a database, providing a seamless solution for dietary tracking. This paper also highlights the potential of real-time food recognition via mobile applications, allowing users to track their food intake conveniently on the go.

In conclusion, this paper demonstrates the significant potential of AI-driven food recognition systems in transforming how we monitor our diets and assess food safety. By automating the identification of food items and calculating nutritional content; these technologies offer a practical, efficient, and non-invasive approach to dietary management. While challenges remain, particularly in dataset diversity and model performance, we are optimistic that future developments in AI will overcome these limitations, driving innovations in personalized nutrition and food safety.

## 2. Dataset

The project's first phase involved conducting an extensive literature review on food classification, segmentation, and calorie estimation techniques, which requires a dataset of food images and calories relative to them.

### 2.1. Dataset Selection

We are selecting the Food-101 dataset https://www.kaggle.com/datasets/dansbecker/food-101, which contains over 100,000 labeled food images spread across 101 food categories for training and testing our models. and for calorie estimation, we used Calories in Food Items (per 100 grams) https://www.kaggle.com/datasets/kkhandekar/calories-in-food-items-per-100-grams, this dataset that will help caloric count and other facts about common foods.

### 2.2. Dataset Analysis and EDA

Exploratory data analysis (EDA) was performed to understand the distribution of food categories in the dataset

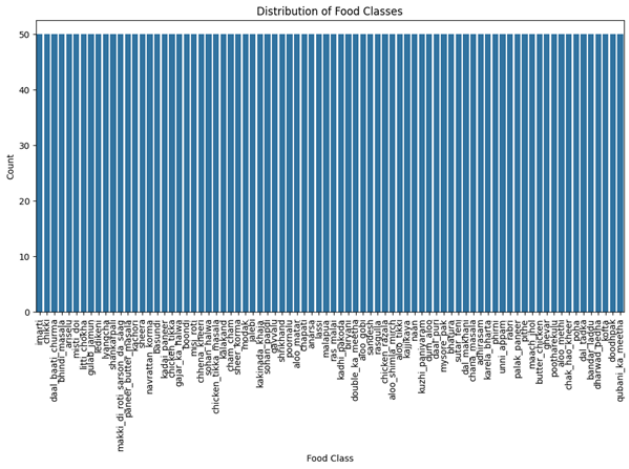


Figure 1. Exploratory data analysis: variability of images across different food types

and to identify potential challenges, such as class imbalances. This analysis provided insights into the variability of images across different food types, and it helped us identify suitable pre-processing techniques for further stages of the project. These images are the dataset sample, which shows the class of its image. This dataset has 80 classes, so I have figured out a few of the images of classes, which is done by using matlplotlib and Seaborn library.


Figure 2. Exploratory data analysis: Image type samples

## 2.3. Data Prepossessing

### 2.3.1 Image data extraction

Data should be processed to extract key features, and all images generally have a high dimensionality which is effective in computation. Feature selection can reduce the number of features to be considered while trying to preserve the essential bits of information. We extracted key features using edge detection and shape and texture detection to identify some features.

Further, the data was prepared to train an image classification model using TensorFlow's Keras API by flattening the input images, which are in a 224 x 224 RGB format, using a flattened layer. Then by splitting the dataset into training and testing sets using a 70/30 split ratio. The train test split function shuffles the data to ensure randomness. It sets a fixed random state for reproducibility after splitting and by defining two image data generators, train gen and test gen, which were responsible for pre-processing the images by rescaling their pixel values between 0 and 1. Additionally, for the training generator, a validation split of 20 percent is specified to set aside a portion of the training data for validation during model training.

And since Keras expects labels to be in string format, the code converts the labels in training and testing sets to string data types, ensuring compatibility with the data generators. This pre-processing step organizes the image dataset efficiently, ready for feeding into a neural network model for training and evaluation.

### 2.3.2 Calorie data extraction

CSV data is opened using Pandas library to load into Data Frame. This step is fundamental as it provides a structured tabular format for analysis and manipulation. The initial dataset exploration allowed us to preview the first few rows, which helped identify the column names and sample values. The dataset's size was determined using the shape function, providing insight into the number of rows and columns. During this pre-processing, unnecessary columns which did not contribute to the predictive task were removed. These preliminary steps ensured a clear understanding of the dataset's structure and highlighted areas requiring cleaning.

## 3. Methodology

The methodology for this project is structured around a step-by-step process that integrates image processing and testing, as well as checking machine learning algorithms to achieve the most accurate food identification and calorie estimation using multiple algorithms. The entire process, from data collection to model evaluation, was designed to ensure the development of a reliable and efficient system.

### 3.1. Current Approach

We approached the classification of images by using the Implementing Multiple models and testing the model's accuracy using the preprocessed data. to identify the best model for the identification of food from the image. and then

### 3.2. Model Creation

#### 3.2.1 For Image Recognization

We started with the Random Forest algorithm to train and test the model, but the accuracy was in the range of 15% to 20% in multiple iterations. Then, we implemented an SVM algorithm using linear and non-linear SVM. The results worsened in accuracy between the range of 5% to 10% in multiple iterations. With the implementation of a combination of models, we were able to increase the accuracy of our model to 30% as the features extracted are not very helpful Now, we implemented a Multi-Layer Perceptron (MLP) model for image classification using TensorFlow's Keras API. The architecture converts the 2D image data into a 1D vector for subsequent dense layers. The model then includes a series of dense (fully connected) layers. The first dense layer consists of 128 neurons with a ReLU activation function, followed by a dropout layer with a 20 percent dropout rate to prevent overfitting. This is followed by another dense layer with 64 neurons, another dropout layer with the same dropout rate, and a final dense layer with 32 neurons using ReLU activation. The output layer consists of a single neuron with a linear activation function, indicating that this model is likely used for regression or classification with one output. The model is then evaluated on a test dataset, and the results show a test loss of 0.030 and an accuracy of 93.37%. These results indicate that the MLP model performs well on the test dataset.

Now, for comparison, we use the best available model to check the Convolutional Neural Network (CNN) for image classification using TensorFlow's Keras API. The create model function builds the model architecture, which includes two convolutional layers with filters of size 16 and 32, followed by max-pooling layers. After this, a global average pooling layer reduces the feature maps to a single vector per image. The model then includes two fully connected dense layers of 64 neurons each, an output layer with specified classes, and a linear activation function. The evaluation results show a test loss of 0.012 and a test accuracy of 98.75 percent. This outcome demonstrates that the model effectively learns and classifies images with high precision on the test data.

#### 3.2.2 For Calories prediction

Implementation of these four machine learning models: Linear Regression, Random Forest Regressor, Support Vector Regressor (SVR), and K-Nearest Neighbors Regressor (KNN). Each model was trained on 80% of the data and evaluated on the remaining 20% using key metrics like Mean Squared Error (MSE), Mean Absolute Error (MAE), and $R^2$ Score. Linear Regression served as the baseline model, while Random Forest Regressor excelled due to its ability to handle non-linear relationships and feature interactions. SVR and KNN also performed well but were sensitive to hyperparameters and feature scaling. The trained models' performances were compared, and the Random Forest Regressor,

### 3.3. Integration of Models

The best-performing model is selected from all models to recognize food from images saved as a pickle file. once a food item is identified, the corresponding nutritional values for that item are extracted or calculated. These values are then passed to the predicted calorie function. The function processes the nutritional inputs, scales them, and uses the Random Forest Regressor to predict the calorie content. This predicted value is then linked to the identified food item, providing the user with an additional layer of information.

## 4. Results

### 4.1. Image recognization Models

#### 4.1.1 Random Forest,SVM, and other combination

Random Forest algorithm did not perform well with the accuracy was in the range of 15% to 20% with multiple iterations. Neither SVM algorithm uses either linear or non-linear SVM. The results worsened in accuracy between the range of 5% to 10%. With the implementation of a combination of models, we were able to increase the accuracy of our model to 30% and was no good

#### 4.1.2 MLP model

Now the observed results for MLP Training vs. Validation Accuracy : The training accuracy starts relatively low and steadily increases across the epochs, reaching approximately 0.9 or 90 percent. In contrast, the validation accuracy remains consistently high, near 1.0 (or 100%) throughout all epochs. This behavior suggests that the model is learning to classify the training data correctly over time while the validation accuracy remains static and very high.

Training vs. Validation Loss : The training loss decreases sharply in the first epoch and then stabilizes, indicating a rapid improvement in model performance. Mean-

```
result = mlp_model.evaluate(test_image, verbose = 0)
print('Test Loss : {:.3f}'.format(result[0]))
print('Test Accuracy : {:.3f}%'.format(result[1]*100))
```

✓ 4.6s

```
Test Loss : 0.030
Test Accuracy : 98.750%
```

Figure 3. MLP Accuracy

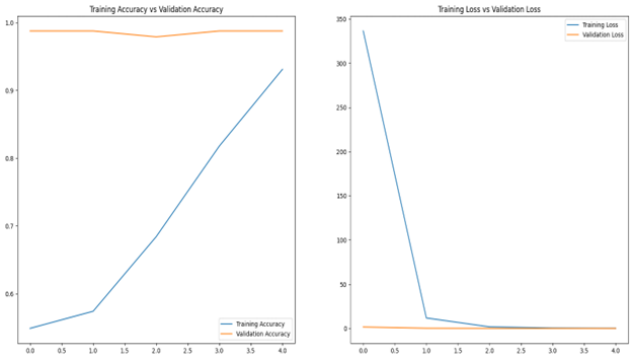


Figure 4. Training and Validation (Loss and Accuracy)

while, the validation loss remains almost constant at a very low value, reflecting minimal errors on the validation dataset.

#### 4.1.3 CNN model

Training vs. Validation Accuracy : The accuracy plot appears constant for both the training and validation sets. The validation accuracy is near 1.0 (or 100 percent), indicating that the model consistently predicts correctly during validation. However, the accuracy of the training remains significantly low or possibly unchanging. This suggests that something might be off with the training process, such as a potential issue with the accuracy calculation or data preparation. Training vs. Validation Loss : The training loss starts at a slightly higher value but quickly converges to a value similar to the validation loss. The validation loss remains constant, suggesting minimal or no overfitting. Still, the drastic drop in training loss within just one epoch raises questions about whether the model genuinely learned from the training data or if there might be an issue with the loss computation.

### 4.2. Calorie predictor Models

Our Four models for calorie predictions were evaluated to predict calorie data.

```
# Evaluate the model on the test set
test_loss, test_accuracy = model.evaluate(test_image, verbose=0)

# Print the results with formatted strings
print(f'Test Loss: {test_loss:.3f}')
print(f'Test Accuracy: {test_accuracy * 100:.3f}%')
```

✓ 7.2s

```
Test Loss: 0.012
Test Accuracy: 98.750%
```

Figure 5. CNN Accuracy

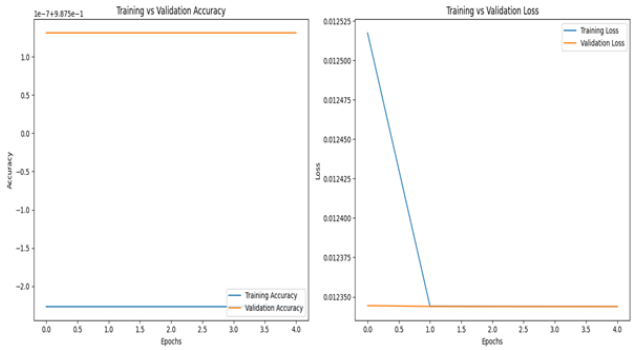


Figure 6. Training and Validation (Loss and Accuracy)

| | MSE | MAE | R2 Score |
|---|---|---|---|
| Linear Regression | 7057.799081 | 62.463062 | 0.744997 |
| Random Forest Regressor | 2675.821597 | 29.849530 | 0.903321 |
| Support Vector Regressor (SVR) | 9795.256237 | 62.826575 | 0.646091 |
| K-Nearest Neighbors Regressor (KNN) | 3663.931445 | 34.312742 | 0.867620 |

Figure 7. Performance Scores

#### 4.2.1 Linear Regression

Linear Regression achieved an MSE of 70.80, MAE of 62.46, and an $R^2$ score of 0.744, indicating moderate performance but limited capacity to handle non-linear relationships.

#### 4.2.2 Random Forest Regressor

The Random Forest Regressor outperformed all models with the lowest MSE (2675.82), MAE (29.85), and the highest $R^2$ score (0.903), showcasing its ability to capture complex patterns and feature interactions effectively.

#### 4.2.3 Support Vector Regressor (SVR)

The Support Vector Regressor struggled with the highest MSE (9795.26), MAE (62.83), and a low $R^2$ score of 0.646, reflecting poor prediction accuracy due to possible hyperparameter tuning issues.

#### 4.2.4 K-Nearest Neighbors Regressor (KNN)

K-Nearest Neighbors provided reasonable performance with an MSE of 3663.93, MAE of 34.31, and an $R^2$ score of 0.867, though it fell short of Random Forest in accuracy. Overall, Random Forest emerged as the best-performing model for this task.

The project successfully developed and integrated a calorie prediction model into a food detection system, providing a comprehensive solution for image-based food recognition and calorie estimation. By combining computer vision techniques with machine learning, the system identifies food items and estimates their caloric content based on nutritional attributes. This integration enhances the functionality of the food detection system, making it a valuable tool for health-conscious users or dietary monitoring applications. The calorie prediction model was built using a robust data preprocessing pipeline, ensuring data consistency and accuracy. Various machine learning models, including Linear Regression, Random Forest Regressor, Support Vector Regressor (SVR), and K-nearest neighbors Regressor (KNN), were implemented and evaluated. Among these, the Random Forest Regressor emerged as the best-performing model with the lowest error metrics and the highest ability to capture data variance. This model was serialized and integrated into the food detection system, enabling seamless calorie predictions based on nutritional inputs such as total fat, cholesterol, sodium, calcium, sugars, saturated fat, serving size, and vitamin C. The integrated system leverages the identified food item's corresponding nutritional features, scales them appropriately, and predicts calorie content using the Random Forest Regressor. This approach ensures scalability, portability, and real-time usability, making it a practical solution for deployment in real-world applications. Overall, this project demonstrates the effectiveness of combining deep learning for food detection with machine learning for calorie estimation, creating a powerful and versatile tool for diet management and health monitoring.

## source

https://github.com/kumarrishav4/Food_calorie_pridictor_ML

## References

[1] Hokuto Kagaya, Kiyoharu Aizawa, and Makoto Ogawa. Food detection and recognition using convolutional neural network. pages 1085–1088, 11 2014. 1

[2] Sushant Kaushal, Dushyanth Kumar Tammineni, Priya Rana, Minaxi Sharma, Kandi Sridhar, and Ho-Hsien Chen. Computer vision and deep learning-based approaches for detection of food nutrients/nutrition: New insights and advances. *Trends in Food Science Technology*, 146:104408, 2024. 1