CALORIE ESTIMATION USING IMAGE PROCESSING AND NEURAL NETWORKS IN MATLAB

Submitted by

MANCHOJU RAVEENA	(21SS1A0443)
S M SAAHIL HUSSAIN	(21SS1A0457)
BAJOJI HARSHITHA	(21SS1A0405)
HASTHAPARAPU BHARGAV KRISHNA	(21SS1A0426)
KAMALAY NAGA SAI	(21SS1A0433)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD UNIVERSITY COLLEGE OF ENGINEERING SULTANPUR

Sultanpur (V), Choutkur (M), Sangareddy (D), Telangana – 502273, India

2024-2025

CALORIE ESTIMATION USING IMAGE PROCESSING AND NEURAL NETWORKS IN MATLAB

PROJECT STAGE - II REPORT SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF BACHELOR OF TECHNOLOGY

IN

ELECTRONICS AND COMMUNICATIONS ENGINEERING

BY

MANCHOJU RAVEENA	(21SS1A0443)
S M SAAHIL HUSSAIN	(21SS1A0457)
BAJOJI HARSHITHA	(21SS1A0405)
HASTHAPARAPU BHARGAV KRISHNA	(21SS1A0426)
KAMALAY NAGA SAI	(21SS1A0433)

UNDER THE GUIDANCE OF

Dr. Y RAGHAVENDER RAO

(Vice Principal & Professor, Electronics and Communication Engineering)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD UNIVERSITY COLLEGE OF ENGINEERING SULTANPUR

Sultanpur (V), Choutkur (M), Sangareddy (D), Telangana – 502273, India

2024-2025

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD UNIVERSITY COLLEGE OF ENGINEERING SULTANPUR

Sultanpur (V), Choutkur (M), Sangareddy (D), Telangana – 502273, India

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

CERTIFICATE

This is to certify that Project Stage - II report work entitled ***"CALORIE ESTIMATION USING IMAGE PROCESSING AND NEURAL NETWORKS IN MATLAB"*** *is a bonafide work carried out by* ***Manchoju Raveena***, ***S M Saahil Hussain***, ***Bajoji Harshitha***, ***Hasthaparapu Bhargav Krishna*** *and* ***Kamalay Nagasai*** *bearing Roll Nos.****21SS1A0443***, ***21SSS1A0457***, ***21SS1A0405***, ***21SS1A0426*** *and* ***21SS1A0433*** *respectively in partial fulfilment of the requirements for the degree of* ***BACHELOR OF TECHNOLOGY*** *in* ***ELECTRONICS & COMMUNICATION ENGINEERING*** *by the Jawaharlal Nehru Technological University, Hyderabad during the academic year 2024- 2025.*

The results embodied in this report have not been submitted to any other University or Institution for the award of any degree or diploma.

----------	----------
Dr. Y. RAGHAVENDER RAO	**Sri. V. RAJANESH**
(Vice Principal & Professor, ECE)	(Associate Professor &
(Project Guide)	Head of the Department, ECE)

ACKNOWLEDGMENT

We take this opportunity to exhibit our deep gratitude to all those who have extended their help by every means possible for the successful completion of this project.

We are grateful to **Dr. G. NARSIMHA**, Principal, JNTUH University College of Engineering, Sultanpur, for his support while pursuing this project.

We would like to thank our guide **Dr. Y. RAGHAVENDER RAO** Vice Principal & Professor, Electronics and Communication Engineering

We thank **Sri. V. RAJANESH** Associate Professor & Head of the Department, Electronics and Communication Engineering for his constant support for his timely advice and valuable suggestions while working on this project as well as throughout the B. Tech course. He has helped to keep us focused and pointed in the right direction.

Finally, we would like to express our gratitude to **Mrs. K SNEHALATHA,** Assistant Professor(c), Department of Electronics and Communication Engineering for spending her valuable time and providing her guidance throughout the course period.

We are grateful to the Project Review Committee members and Department of Electronics and Communication Engineering who have helped in successfully completing this project by giving their valuable suggestions and support.

We thank our friends, families and all professors of ECE Department who have supported us all the way during the project.

MANCHOJU RAVEENA	(21SS1A0443)
S M SAAHIL HUSSAIN	(21SS1A0457)
BAJOJI HARSHITHA	(21SS1A0405)
HASTHAPARAPU BHARGAV KRISHNA	(21SS1A0426)
KAMALAY NAGA SAI	(21SS1A0433)

ABSTRACT

Obesity, a serious chronic disease, is on the rise as a result of how easily food can be brought to our door steps. People's need for food grew, and at the same time, their anxiety about their nutrition also grew. In recent years, there has been a growing interest in automating dietary assessment to facilitate healthier eating habits and personalized nutrition plans.

This project presents an innovative approach to nutritional analysis and calorie estimation through image processing techniques combined with deep learning methodologies. This study offers an image-based calorie estimation system that asks the user to upload an image of a food item in order to calculate the estimated number of calories in the image. Utilizing MATLAB's powerful image processing and machine learning capabilities, we developed a robust framework that leverages convolutional neural networks (CNNs) to accurately identify and classify food items from images. The model is trained on a comprehensive dataset containing diverse food images along with their nutritional information, enabling it to predict calorie counts and nutrient compositions effectively.

The proposed system demonstrates high accuracy in food recognition and provides real-time calorie estimation, thereby assisting users in making informed dietary choices. By integrating this technology into mobile applications, we aim to enhance user engagement with nutritional information, ultimately promoting healthier lifestyles and improving public health outcomes.

KEYWORDS :

Automated Nutritional Analysis, Calorie Estimation, Deep Learning, Image Processing, Dietary Assessment, Convolutional Neural Networks (CNN), Food Recognition, MATLAB, Real-time Analysis, Mobile Applications, Nutritional Information, Personalized Nutrition.

INDEX

TABLE OF FIGURES

1. INTRODUCTION

1.1 INTRODUCTION

Calorie estimation using image processing and Convolutional Neural Networks (CNNs) is a modern approach to automate dietary assessment by analyzing food images. With the rise of health-conscious lifestyles and the need for accurate nutritional tracking, this technique leverages computer vision and deep learning to identify food items and estimate their caloric values.

MATLAB provides a robust platform for implementing such systems, offering powerful tools for image processing, deep learning, and machine learning. CNNs, a type of deep learning model, play a crucial role in classifying food items by learning patterns from large datasets. When combined with image processing techniques, CNNs enable accurate identification, segmentation, and volume estimation of food items, leading to reliable calorie computation.

In this project we have proposed a model which is focused on Convolutional Neural Network (CNN) for food detection from a given image. Initially, we have developed a dataset of 23 food categories for identifying multiple food items with the help of [1] and [2]. Then we extracted features from the images and trained the model with CNN. Next, we deployed the model in MATLAB which is for showing the result. In the end, the result is evaluated in the MATLAB when an image is given as an input and the system predicts what food it is along with showing the calorie amount.

1.2 AIM

The primary aim of this project is to develop an **automated and accurate calorie estimation system** using **image processing and Convolutional Neural Networks (CNNs) in MATLAB**. This system will analyze food images, classify different food items, estimate portion sizes, and compute their corresponding calorie content.Specific objectives involves Enhancing the image quality and segment food items using MATLAB's Image Processing Toolbox, utilize CNNs to accurately classify different food items from images,

to develop techniques to estimate the volume or weight of food items for accurate calorie computation and to Integrate a nutritional database to estimate calorie content based on identified food items and portion sizes also to design an interactive MATLAB-based system that can assist users in monitoring their food intake conveniently.

1.3 METHODOLOGIES

There are various phases involved in this process of calorie estimation**. Image Processing phase, Food Classification, Performance Evaluation and Testing phase.**

Image Processing Phase

The **image processing phase** is crucial in preparing food images for analysis, ensuring accurate classification and calorie estimation. This phase involves several steps to enhance image quality, remove noise, and segment food items from the background.

Key Steps in the Image Processing Phase

1. Image Acquisition

Capturing food images using a camera or obtaining them from a dataset. Ensuring consistent lighting and angles for better recognition.

2. Image Preprocessing

Resizing: Standardizing image dimensions for CNN input (e.g., 224x224 pixels for models like AlexNet).

Noise Reduction: Applying filters (Gaussian, median) to remove unwanted distortions.

Color Normalization: Adjusting brightness and contrast for uniform image quality.

3. Image Segmentation

Thresholding: Separating food items from the background using color or intensity thresholds.

Edge Detection: Using techniques like Canny or Sobel edge detection to identify object boundaries.

Contour Detection: Extracting distinct food items from the plate.

4. Feature Extraction

Identifying shape, texture, and color features for better classification.

Extracting edges, histograms, and keypoints using MATLAB's **Image Processing Toolbox**.

This phase ensures that the input images are optimized for deep learning analysis, leading to better food classification, portion estimation, and calorie computation.

Food Classification Phase in Calorie Estimation

The food classification phase is responsible for identifying different food items from images using Convolutional Neural Networks (CNNs). This step is crucial because accurate classification directly affects portion estimation and calorie computation.

Key Steps in Food Classification

1. Dataset Preparation

Collect labeled food images from datasets like Food-101, UEC-Food100, or custom datasets. Split the dataset into training, validation, and testing sets for model evaluation. Apply data augmentation (rotation, flipping, scaling) to improve model robustness.

2. CNN Model Selection and Training

Use pretrained CNN models like AlexNet, VGG-16, ResNet or train a custom CNN. Fine-tune the model using transfer learning in MATLAB's Deep Learning Toolbox. Optimize parameters like learning rate, batch size, and number of epochs.

3. Food Item Classification

Input processed food images into the trained CNN model. Extract deep features and classify food into predefined categories. Output the recognized food item with a confidence score. This phase ensures that the system can correctly estimate portion sizes and compute nutritional values.

Performance Evaluation Phase

The **performance evaluation phase** assesses the accuracy and reliability of the food classification and calorie estimation system. This step ensures that the model performs well in real-world scenarios and provides accurate results.

Key Metrics for Evaluation

1. Classification Accuracy

Measures how well the CNN model identifies different food items.

Formula:

Image **Accuracy = (Correct Predictions)/(Total Predictions) × 100**

2. Confusion Matrix

Evaluates the classification model by showing correct and incorrect predictions for each food category. Helps in identifying misclassifications.

Testing and Validation Phase

The system is tested on **real-world food images** to validate its performance. Performance is compared across different CNN models (e.g., AlexNet, ResNet).

1.4 ORGANIZATION OF THESIS

The structure of this thesis is as follows:

- Chapter 1: **Introduction** This chapter provides an overview of calorie estimation in matlab, outlines the aim and methodology of the project, and highlights its significance.
- Chapter 2: **Literature Review** This chapter reviews existing research and developments in Food calorie estimation using various methodologies, focusing on historical milestones, challenges, and emerging trends.
- Chapter 3: **System Design and Implementation** This chapter details the design and implementation of food calorie estimation in MATLAB, including CNN techniques and food classification.
- Chapter 4: **Results and Analysis** This chapter presents the results of system testing, evaluates its performance, and discusses findings in the context of existing technologies.
- Chapter 5: **Conclusion and Future Work** The final chapter summarizes the project's contributions, discusses its limitations, and suggests directions for future research.

By following this structure, the thesis aims to provide a comprehensive exploration of Food calorie estimation and its potential to enhance accessibility.

2. LITERATURE SURVEY

2.1 FAST FOOD RECOGNITION FROM VIDEOS OF EATING FOR CALORIE ESTIMATION [1]

Accurate and passive acquisition of dietary data from patients is essential for a better understanding of the etiology of obesity and development of effective weight management programs. Self-reporting is currently the main method for such data acquisition. However, studies have shown that data obtained by self-reporting seriously underestimate food intake and thus do not accurately reflect the real habitual behavior of individuals. Computer food recognition programs have not yet been developed. In this paper, we present a study for recognizing foods from videos of eating, which are directly recorded in restaurants by a web camera. From recognition results, our method then estimates food calories of intake. We have evaluated our method on a database of 101 foods from 9 food restaurants in USA and obtained promising results.

This paper proposes and studies methods which recognize fast foods from videos of eating and estimate meal calories based on recognizing foods. Our work shows that an off-the shelf vision algorithm with appropriate engineering can lead to promising results on this interesting camera-based task. But our work does not provide any observation or conclusion between obesity and fast foods

2.2 A VIDEO PROCESSING APPROACH TO THE STUDY OF OBESITY [2]

Currently, a significant obstacle in identifying the most important factors that contribute to obesity is the lack of appropriate tools to evaluate food consumption and physical activity on a daily basis. This paper presents a novel video-based approach to estimating the energy intake and expenditure of individuals who are over-weight or obese. We propose to utilize a miniature video camera to chronically record data surrounding an individual's daily activity. From the recorded data, energy intake and expenditure are extracted objectively using signal processing algorithms. We discuss two algorithms which are both featured with low computational complexity, suitable for processing large sets of field-acquired

video data. These algorithms estimate: (i) energy intake (caloric content) by determining the physical dimensions of the foods consumed; and (ii) energy expenditure by evaluating the extent of the subject's physical activity. We also report initial experimental results to demonstrate the effectiveness of the new approach.

2.3 FOOD RECOGNITION USING STATISTICS OF PAIRWISE LOCAL FEATURES [3]

Food recognition is difficult because food items are deformable objects that exhibit significant variations in appearance. We believe the key to recognizing food is to exploit the spatial relationships between different ingredients (such as meat and bread in a sandwich). We propose a new representation for food items that calculates pairwise statistics between local features computed over a soft pixel level segmentation of the image into eight ingredient types. We accumulate these statistics in a multi-dimensional histogram, which is then used as a feature vector for a discriminative classifier.

Our experiments show that the proposed representation is significantly more accurate at identifying food than existing methods. Food recognition is a new but growing area of exploration. Although many of the standard techniques developed for object recognition are ill-suited to this problem, we argue that exploiting the spatial characteristics of food, in combination with statistical methods for pixel-level image labeling will enable us to develop practical systems for food recognition.

2.4 COMBINING GLOBAL AND LOCAL FEATURES FOR FOOD IDENTIFICATION IN DIETARY ASSESSMENT [4]

Many chronic diseases, such as heart diseases, diabetes, and obesity, can be related to diet. Hence, the need to accurately measure diet becomes imperative. We are developing methods to use image analysis tools for the identification and quantification of food consumed at a meal. In this paper we describe a new approach to food identification using several features based on local and global measures and a “voting” based late decision fusion classifier to identify the food items. Experimental results on a wide variety of food items are presented.

2.5 A FOOD RECOGNITION SYSTEM FOR DIABETIC PATIENTS BASED ON AN OPTIMIZED BAG-OF-FEATURES MODEL [5]

Computer vision-based food recognition could be used to estimate a meal's carbohydrate content for diabetic patients. This study proposes a methodology for automatic food recognition, based on the bag-of-features (BoF) model. An extensive technical investigation was conducted for the identification and optimization of the best performing components involved in the BoF architecture, as well as the estimation of the corresponding parameters. For the design and evaluation of the prototype system, a visual dataset with nearly 5000 food images was created and organized into 11 classes. The optimized system computes dense local features, using the scale-invariant feature transform on the HSV color space, builds a visual dictionary of 10000 visual words by using the hierarchical k-means clustering and finally classifies the food images with a linear support vector machine classifier. The system achieved classification accuracy of the order of 78%, thus proving the feasibility of the proposed approach in a very challenging image dataset.

The first experiment proved the superiority of dense key point extraction which was able to produce the required large number of patches with minimum overlap between them. The second experiment investigated the effect of the descriptor's size on the final performance. The best results were obtained by the combination of descriptors with sizes 16, 24, and 32. By using descriptors with different sizes, the BoF system gained multi resolution properties that increased the final performance, since the food scale may vary among the images. Then, the hsvSIFT was chosen among 14 different color and texture descriptors as giving the best results. hsvSIFT constitutes a differential descriptor that describes the local texture in all three different color channels of the HSV color space. This fact enables it to include color information, apart from texture, but also keep some invariance in intensity and color changes. The color capturing ability of hsvSIFT was also proved by the descriptors' fusion experiment that failed to increase the performance after combining it with the best color descriptors. As regards the learning of the visual dictionary, k-means was compared to its hierarchical version hk-means. The latter managed to produce almost equivalent results with k-means, for the optimal number of visual words, while being extremely

computationally efficient. The optimal number of words was determined to be approximately 10000, since fewer words resulted in clearly worse results and more words did not improve the performance. For the final classification, two linear and four nonlinear machine-learning techniques were employed, with the linear giving the best results, especially for large number of features. This is probably caused by the high dimensionality of the feature space, as this makes the problem linearly separable, at least to some extent.

2.6 CLASSIFYING FOOD IMAGES REPRESENTED AS BAG OF TEXTONS [7]

The classification of food images is an interesting and challenging problem since the high variability of the image content which makes the task difficult for current state-of-the-art classification methods. The image representation to be employed in the classification engine plays an important role. We believe that texture features have been not properly considered in this application domain. This paper points out, through a set of experiments, that textures are fundamental to properly recognize different food items. For this purpose the bag of visual words model (BoW) is employed. Images are processed with a bank of rotation and scale invariant filters and then a small codebook of Textons is built for each food class. The learned class-based Textons are hence collected in a single visual dictionary. The food images are represented as visual words distributions (Bag of Textons) and a Support Vector Machine is used for the classification stage. The experiments demonstrate that the image representation based on Bag of Textons is more accurate than existing (and more complex) approaches in classifying the 61 classes of the Pittsburgh Fast-Food Image Dataset.

This paper proposes food image classification methods exploiting both local appearance and global structural information of food objects. The contribution of the paper is threefold. First, non-redundant local binary pattern (NRLBP) is used to describe the local appearance information of food objects. Second, the structural information of food objects is represented by the spatial relationship between interest points and encoded using a shape context descriptor formed from those interest points. Third, we propose two methods of integrating appearance and structural information for the description and classification of food images. We evaluated the proposed methods on two datasets. Experimental results

verified that the combination of local appearance and structural features can improve classification performance.

The pervasiveness of mobile cameras has resulted in a dramatic increase in food photos, which are pictures reflecting what people eat. In this paper, we study how taking pictures of what we eat in restaurants can be used for the purpose of automating food journaling. We propose to leverage the context of where the picture was taken, with additional information about the restaurant, available online, coupled with state-of-the-art computer vision techniques to recognize the food being consumed. To this end, we demonstrate image-based recognition of foods eaten in restaurants by training a classifier with images from restaurant's online menu databases. We evaluate the performance of our system in unconstrained, real-world settings with food images taken in 10 restaurants across 5 different types of food (American, Indian, Italian, Mexican and Thai).

2.7 FOOD RECOGNITION AND LEFTOVER ESTIMATION FOR DAILY DIET MONITORING [10]

In this paper, we apply a convolution neural network (CNN) to the tasks of detecting and recognizing food images. Because of the wide diversity of types of food, image recognition of food items is generally very difficult. However, deep learning has been shown recently to be a very powerful image recognition technique, and CNN is a state-of-the-art approach to deep learning. We applied CNN to the tasks of food detection and recognition through parameter optimization. We constructed a dataset of the most frequent food items in a publicly available food-logging system, and used it to evaluate recognition performance. CNN showed significantly higher accuracy than did traditional support-vector-machine-based methods with handcrafted features. In addition, we found that the convolution kernels show that color dominates the feature extraction process. For food image detection, CNN also showed significantly higher accuracy than a conventional method did.

We report the feature obtained from the Deep Convolutional Neural Network boosts food recognition accuracy greatly by integrating it with conventional hand-crafted image features, Fisher Vectors with HoG and Color patches. In the experiments, we have achieved 72.26% as the top-1 accuracy and 92.00% as the top-5 accuracy for the 100-class food

dataset, UEC-FOOD100, which outperforms the best classification accuracy of this dataset reported so far, 59.6%, greatly.

Food image recognition is one of the promising applications of visual object recognition in computer vision. In this study, a small-scale dataset consisting of 5822 images of ten categories and a five-layer CNN was constructed to recognize these images. The bag-of-features (BoF) model coupled with support vector machine was first tested as comparison, resulting in an overall accuracy of 56%; while the CNN performed much better with an overall accuracy of 74%. Data expansion techniques were applied to increase the size of training images, which achieved a significantly improved accuracy of more than 90% and prevent the overfitting issue that occurred to the CNN without using data expansion. Further improvement is within reach by collecting more images and optimizing the network architecture and relevant hyper-parameters.Food image recognition is one of the promising applications of visual object recognition in computer vision. In this study, a small-scale dataset consisting of 5822 images of ten categories and a five-layer CNN was constructed to recognize these images.

2.8 CONCLUSION

Both research works contribute significantly to automated food recognition and calorie estimation using deep learning. While Prof. Subhi et al.'s approach focused on CNN-based classification, Prof. Ayon et al. leveraged Inception V3 for better feature extraction and multi-food detection.

However, challenges remain, particularly in portion size estimation, computational complexity, and distinguishing visually similar food items. Future advancements in dataset expansion, 3D reconstruction, and real-time implementation will enhance the effectiveness of such systems for dietary tracking and health monitoring.

3. BASICS OF IMAGE PROCESSING

3.1 IMAGE

An image is a two-dimensional (2D) representation of the three-dimensional (3D) world. It can be mathematically represented as **I(x, y)**, where **(x, y)** denotes the position of a pixel in the image, and **I** represents the intensity or brightness value at that specific pixel location. This intensity value determines the visual appearance of the image at that point.

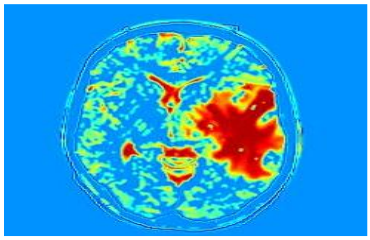

Figure 3.1 An example of an RGB image

Black and White image (Binary/monochrome image)

The black and white image is having only two possible pixel values (0/1) As it contain two (0/1) pixel values so called Binary Image. **1 pixel = 1 bit (0/1)** As each pixel contain only one colour (either black or white) so called Monochrome.

1 ⟶ White

0 ⟶ Black

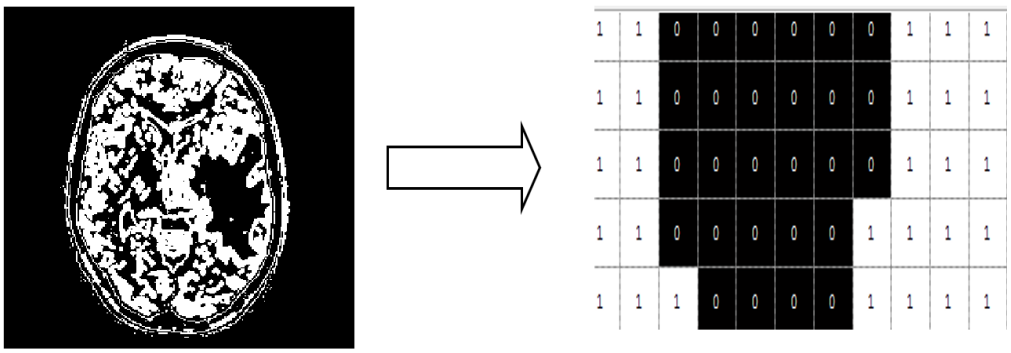

Figure 3.2 monochrome image representation in pixels

Gray-scale Image(scale of 0-255)

The gray scale image is having pixel value from 0 to 255. This is called as gray scale (scale of 0 to 255). If we see the black and images, we are not getting some parameters. So we used color map called as shades of black and white **1 pixel = 1 byte (8 bits)** So using gray images we are getting good parameters as well as operations are easier compared to colour images.

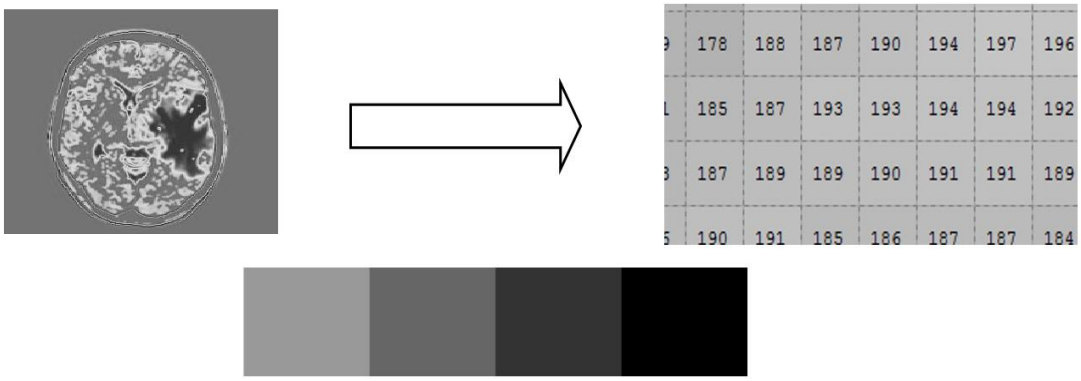

Figure 3.3 grayscale image and different intensities

Colour Images (RGB images)

Each colour in universe is made by mixing Red, green and blue colour **1 pixel = 3 bytes (24 bits)**So, RGB image is having pixel values between 0-255.But each pixel contains 3 colors(R, G and B).

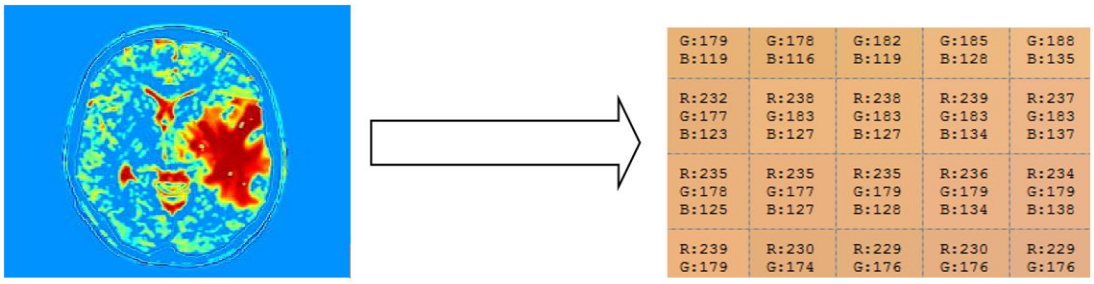

Figure 3.4 RGB image and its pixel representation

3.2 IMAGE PROCESSING

Analysis and manipulation on image to get some feature or to improve the quality of that image. E.g. detection and tracking, Removal of noise from blur image.

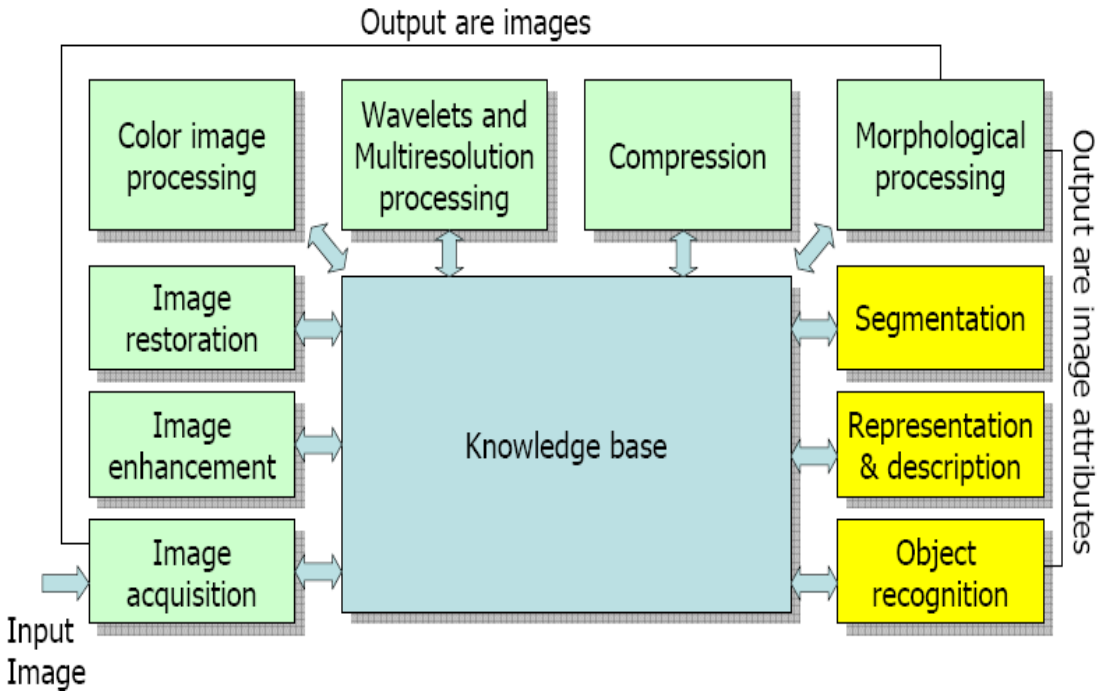

Figure 3.5 components of digital image processing

1. Image Acquisition

The first stage of any vision system is the image acquisition stage. Image acquisition is nothing but capturing of an image using some reflecting phenomenon. After the image has been obtained, various methods of processing can be applied to the image to perform the many different vision tasks required today.

2. Image Enhancement

The principal objective of image enhancement is to process a given image so that the result is more suitable than the original image for a specific application. It accentuates or sharpens image features such as edges, boundaries, or contrast to make a graphic display more helpful for display and analysis. The enhancement doesn't increase the inherent information content of the data, but it increases the dynamic range of the chosen features so that they can be detected easily. The greatest difficulty in image enhancement is quantifying the criterion for enhancement and therefore, a large number of image enhancement techniques are empirical and require interactive procedures to obtain satisfactory results. Image enhancement methods can be based on either spatial or frequency domain techniques.

3. Image Restoration

The purpose of image restoration is to "compensate for" or "undo" defects which degrade an image. Degradation comes in many forms such as motion blur, noise, and camera miss focus. In cases like motion blur, it is possible to come up with a very good estimate of the actual blurring function and "undo" the blur to restore the original image. In cases where the image is corrupted by noise, the best we may hope to do is to compensate for the degradation it caused. In this project, we will introduce and implement several of the methods used in the image processing world to restore images.

4. Color Image Processing

The human visual system can distinguish hundreds of thousands of different colour shades and intensities, but only around 100 shades of grey. Therefore, in an image, a great deal of extra information may be contained in the colour, and this extra information can then be used to simplify image analysis, e.g. object identification and extraction based on colour. Three independent quantities are used to describe any particular colour. The hue is determined by the dominant wavelength. Visible colours occur between about 400nm (violet) and 700nm (red) on the electromagnetic spectrum.

5. Wavelets and Multiresolution Processing

Unlike the Fourier transform, which decomposes a signal to a sum of sinusoids, the wavelet transform decomposes a signal (image) to small waves of varying frequency and limited duration. The advantage is that we also know when (where) the frequencies appear. We will examine wavelets from a multi resolution point of view and begin with an overview of imaging techniques involved in multi resolution theory.

In numerical analysis and functional analysis, a discrete wavelet transform (DWT) is any wavelet transform for which the wavelets are discretely sampled. As with other wavelet transforms, a key advantage it has over Fourier transforms is temporal resolution: it captures both frequency and location information (location in time).

7. Image Compression

There are two categories of compression techniques used with digital graphics, lossy and lossless. Whilst each uses different techniques to compress files, both have the same aim. To look for duplicate data in the graphic and use a much more compact data representation. Lossy and Lossless each have various methods which are used by different file formats and achieve different results. Therefore not all lossy or lossless formats will use the same methods. It is beyond the scope of this Unit to look at these methods in detail so you will not be assessed on them. The Unit entitled Digital Imaging: Bitmaps covers compression methods in more detail. If you are a bit unclear about this, the following may help:

Lossy compression methods include DCT (Discreet Cosine Transform), Vector Quantization and Huffman coding.

Lossless compression methods include RLE (Run Length Encoding), string-table compression, LZW (Lempel Ziff Welch) and zlib.

8. Morphological Image Processing

Morphological image processing is a collection of non-linear operations related to the shape or morphology of features in an image. According to Wikipedia, morphological operations rely only on the relative ordering of pixel values, not on their numerical values, and therefore are especially suited to the processing of binary images. Morphological operations can also be applied to grayscale images such that their light transfer functions are unknown and therefore their absolute pixel values are of no or minor interest. Morphological techniques probe an image with a small shape or template called a structuring element. The structuring element is positioned at all possible locations in the image and it is compared with the corresponding neighbourhood of pixels.

It includes following four operations:

1. Dilation
2. Erosion
3. Opening
4. Closing

9. Image Segmentation

In computer vision, image segmentation is the process of partitioning a digital image into multiple segments (sets of pixels, also known as super pixels). The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze. Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images. More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain characteristics.

The result of image segmentation is a set of segments that collectively cover the entire image, or a set of contours extracted from the image (see edge detection). Each of the pixels in a region is similar with respect to some characteristic or computed property, such as color, intensity, or texture. Adjacent regions are significantly different with respect to the same characteristic(s). When applied to a stack of images, typical in medical imaging, the resulting contours after image segmentation can be used to create 3D reconstructions with the help of interpolation algorithms like marching cubes.

10. Image Representation, Object Recognition

Object recognition is a process for identifying a specific object in a digital image or video. Object recognition algorithms rely on matching, learning, or pattern recognition algorithms using appearance-based or feature-based techniques. After an image is segmented into regions; the resulting aggregate of segmented pixels is represented & described for further computer processing. Representing region involves two choices: in terms of its external characteristics (boundary) in terms of its internal characteristics (pixels comprising the region).Above scheme is only part of task of making data useful to computer. Next task is to describe the region based on representation. Ex. A region may be represented by its boundary & its boundary is described by features such as its length, the orientation of straight line joining its extreme points & number of concavities in the boundary.

3.3 CONCLUSION

In this chapter, we explored the foundational concepts of digital images and the key processes involved in image processing. Starting with the types of images—binary, grayscale, and color—we understood how visual information is represented and interpreted

by computers using pixel intensity values. We then delved into various stages of digital image processing, including image acquisition, enhancement, restoration, and segmentation. Advanced techniques such as color image processing, wavelets, and morphological operations were also discussed, each offering powerful tools for improving and analyzing image data. Additionally, we covered crucial applications like image compression and object recognition, highlighting their importance in real-world scenarios such as medical imaging, computer vision, and AI systems. Overall, this chapter laid the groundwork for understanding how digital images are manipulated to extract meaningful information, forming the basis for numerous practical applications across industries.

4. CONVOLUTIONAL NEURAL NETWORKS

4.1 NEURAL NETWORKS

The term NN is very general and it describes broad family of models. In this context NN is distributed and parallel model that is capable of approximating complex nonlinear functions. Network is composed from multiple computational units called neurons assembled in particular topology. Description of NN structure will follow the convention laid out in the description of learning algorithm. Meaning that a description of the learning algorithm is composed of model, cost function and optimization procedure. The difference comes into play with the fact that model of NN is much more complex than the model linear regression. Therefore, the analysis is divided into model of neuron and topology of the network.

Feed forward neural network or Multilayer Perceptron with multiple hidden layers in artificial neural networks is usually known as Deep Neural Networks (DNNs). Convolutional Neural Networks (CNN) is one kind of feed forward neural network. In 1960s, when Hubel and Wiesel researched the neurons used for local sensitive orientation-selective in the cat's visual system, they found the special network structure can effectively reduce the complexity of Feedback Neural Networks and then proposed Convolution Neural Network. CNN is an efficient recognition algorithm which is widely used in pattern recognition and image processing. It has many features such as simple structure, less training parameters and adaptability. It has become a hot topic in voice analysis and image recognition. Its weight shared network structure make it more similar to biological neural networks. It reduces the complexity of the network model and the number of weights.

Generally, the structure of CNN includes two layers one is feature extraction layer, the input of each neuron is connected to the local receptive fields of the previous layer, and extracts the local feature. Once the local features are extracted, the positional relationship between it and other features also will be determined. The other is feature map layer, each computing layer of the network is composed of a plurality of feature map. Every feature map is a plane, the weight of the neurons in the plane are equal. The structure of feature map uses the sigmoid function as activation function of the convolution network, which

makes the feature map have shift invariance. Besides, since the neurons in the same mapping plane share weight, the number of free parameters of the network is reduced. Each convolution layer in the convolution neural network is followed by a computing layer which is used to calculate the local average and the second extract, this unique two feature extraction structure reduces the resolution. CNN is mainly used to identify displacement, zoom and other forms of distorting invariance of two-dimensional graphics. Since the feature detection layer of CNN learns by training data, it avoids explicit feature extraction and implicitly learns from the training data when we use CNN. Furthermore, the neurons in the same feature map plane have the identical weight, so the network can study concurrently.

This is a major advantage of the convolution network with respect to the neuronal network connected to each other. Because of the special structure of the CNN's local shared weights makes it have a unique advantage in speech recognition and image processing. Its layout is closer to the actual biological neural network. Shared weights reduces the complexity of the network. In particular multi-dimensional input vector image can directly enter the network, which avoids the complexity of data reconstruction in feature extraction and classification process. Face recognition is a biometric identification technology based on the facial features of persons. The study of face recognition system began in the 1960s, in the late 1980s with the development of computer technology and optical imaging techniques it has been improved; in the late 1990s it truly entered the stages of initial applications. In practical applications, such as monitoring system, the collected face images captured by cameras are often low resolution and with great pose variations. Affected by pose variation and low resolution, the performance of face recognition degrades sharply. And pose variations bring great challenge to face recognition.

They bring nonlinear factors into face recognition. And some of the existing machine learning method mostly use shallow structure. Deep learning can achieve the approximation of complex function by a deep nonlinear network structure. In this article, we use convolution neural network to solve face recognition. It can overcome the influence of pose or resolution in face recognition. Due to the long training time and the large recognition computing, it is difficult to meet the real-time requirements, or the delay

exceeds the range of tolerance. So we use the cloud platform to concurrently speed up the computing process.

Topology of the Network

There are several different commonly used topologies. Two most commonly used in deep learning are feed-forward and recurrent. Feed forward networks are characterized by the fact that during activation the information flow only in forward

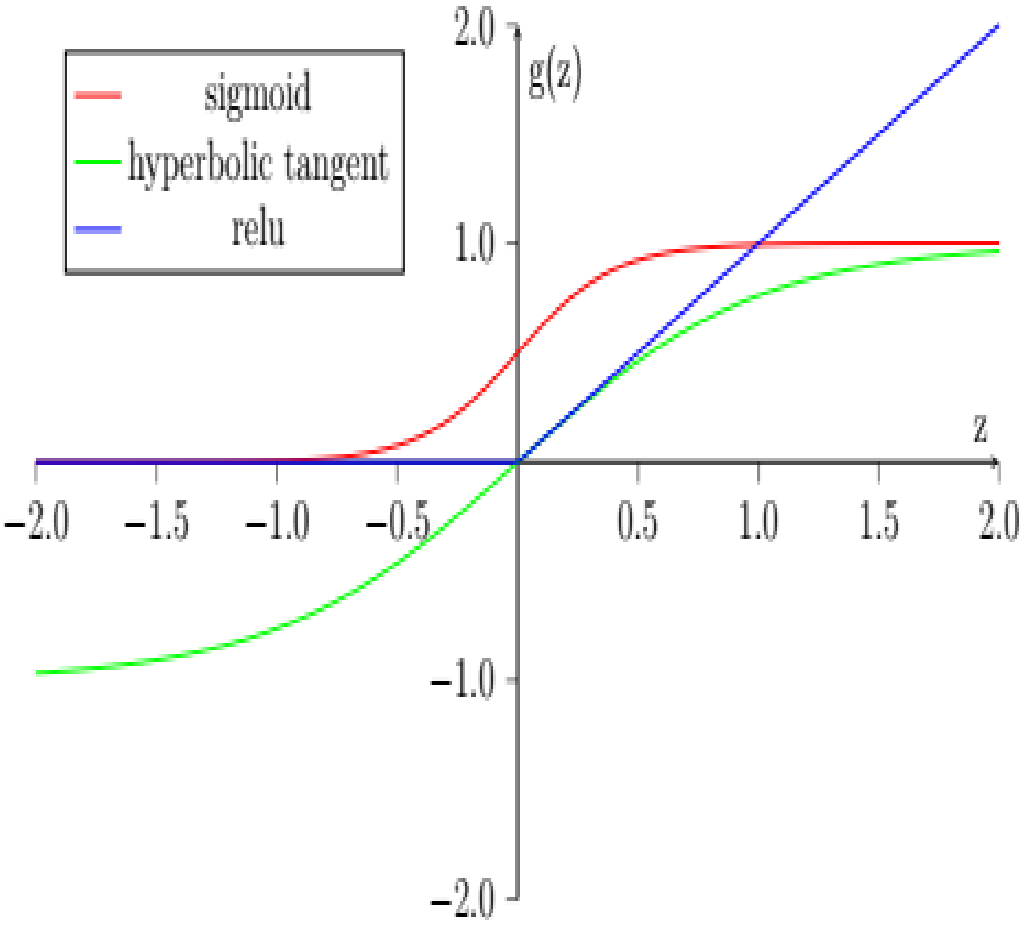

Figure 4.1 activation functions

direction from inputs to output. A recurrent network has some sort of feedback loop. Another criterion of topology is how are individual neurons in the network connected. Most commonly are NNs ordered in layers. In each layer there can be from 1 to n neurons. Layers are hierarchically stacked. In typical terminology the frst layer is called input layer, the last

layer is called output layer and the layers in-between are called hidden. Description of the network rests on interconnections between individual layers. Most common scheme is called fully connected where each neuron in hidden layer has input connections from all neurons from previous layer − 1 and its output is connected to input of each neuron in following + 1 layer. From this point on the term NN will refer to Feed-forward Fully Connected Neural Network. Types of neurons are dependent on the type of the layer. Currently the main difference is in their activation function, which wasn't the case for a long time. Historically all layers had neurons with sigmoid activation function. It was mainly because the output sigmoid layer can be easily mapped onto probability distribution, since it acquires vales between 0 and 1. Only relatively recently2 it was found that network composed of neurons with ReLU activation function in the hidden layers can be trained very quickly and are more resistant against over-ftting. Activation functions are still subject of ongoing research. Neurons in output layer need output that can produce probability distribution.

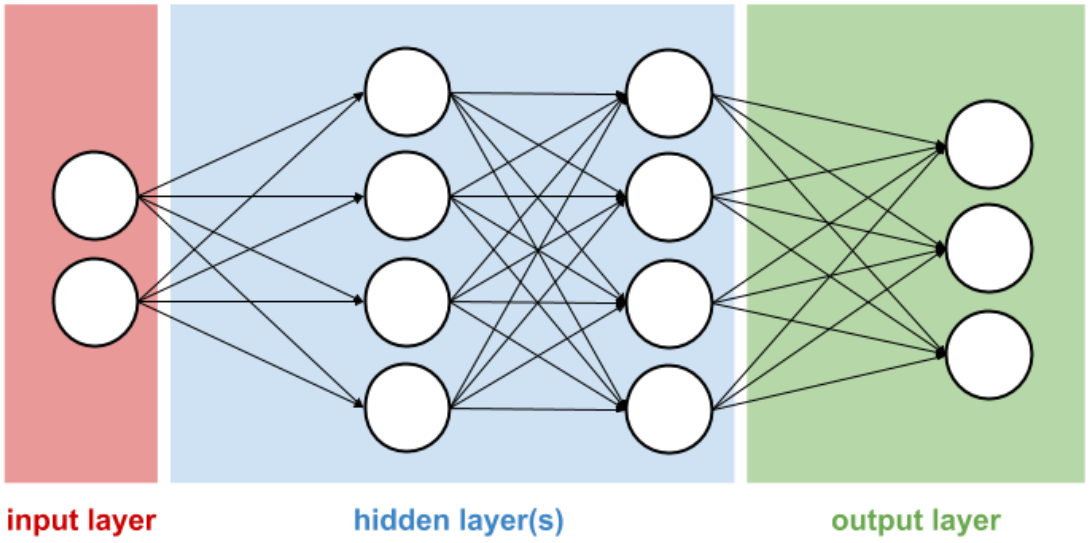

Figure 4.2 Fully connected Feed Forward Neural Network

That can be used to estimate the probability of individual classes. For this reason, most commonly used activation function of output neuron is called softmax. Softmax is normalized exponential function. It is used to represent probability of an instance being member of class j as

$$g(z)_j = \frac{e^{z_j}}{\sum_{K}^{k=1} e^{z_k}},$$

where k is total number of classes.

4.2 CONVOLUTIONAL NEURAL NETWORKS

Convolutional Neural Networks (CNNs) have revolutionized image processing by enabling automatic feature extraction and classification. In the context of calorie estimation, CNNs play a crucial role in identifying different food items from images and mapping them to their respective caloric values. MATLAB provides a comprehensive Deep Learning Toolbox, which simplifies the implementation of CNNs, making it an ideal platform for this task. The process involves multiple stages, including dataset preparation, model definition, training, evaluation, and the final calorie estimation.

The first step in implementing a CNN-based calorie estimation system is to acquire and organize a dataset of food images. This dataset should consist of labeled images of various food items, with each category corresponding to a specific food type. The images need to be stored in separate folders, with each folder representing a unique class. MATLAB's imageDatastore function allows easy management of these datasets, automatically labeling the images based on their folder names. Since deep learning models require large amounts of data for effective training, it is often necessary to augment the dataset by applying transformations such as rotation, flipping, and brightness adjustments. This helps the model generalize better to real-world variations in food appearance.Once the dataset is prepared, it is divided into training and validation subsets. Typically, 80% of the images are used for training, while the remaining 20% are reserved for validation. This split ensures that the model can learn from a diverse set of images while also being tested on unseen data during training. The next step is to define the CNN architecture. A CNN consists of several layers that work together to extract relevant features from the images.

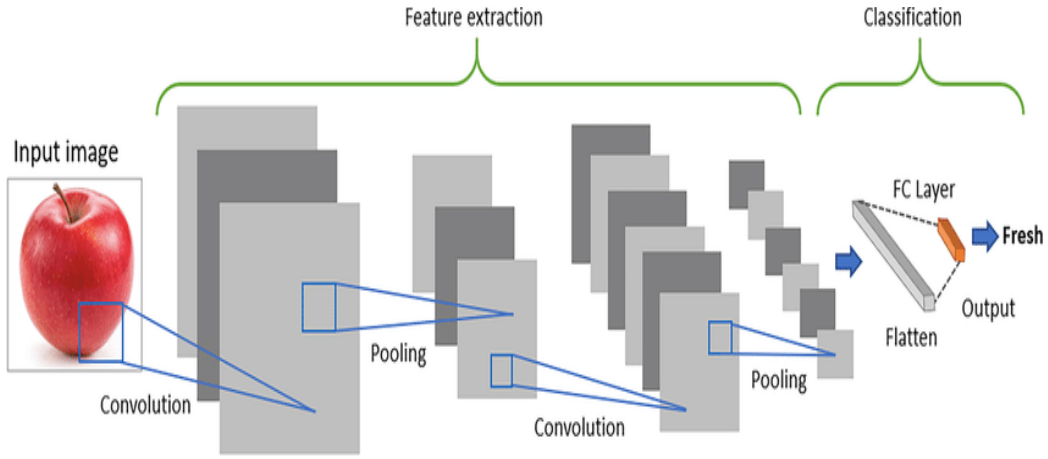

Figure 4.3 Layers of CNN

With the CNN architecture defined, the next step is to configure training options. Training a deep learning model requires careful selection of parameters such as the optimizer, learning rate, batch size, and number of epochs. MATLAB provides various optimization algorithms, with Adam being a popular choice due to its adaptive learning rate properties. The model is then trained using the train Network function, which iteratively updates the network's weights to minimize classification errors. During training, the model's performance is continuously evaluated on the validation dataset to ensure that it is learning effectively. If the validation accuracy is low or the model overfits, techniques such as dropout regularization and batch normalization can be applied to improve generalization.

After training is complete, the model is tested on a separate set of images to evaluate its performance. The classification accuracy is measured by comparing the predicted food labels with the actual labels. If the accuracy is satisfactory, the trained model can be used for real-time food recognition. To classify a new food image, it is first preprocessed to match the input size of the CNN. The model then predicts the food category based on the extracted features. Once the food item is identified, its caloric value is estimated using a predefined nutritional database. This database contains standard calorie values for different food items, which are mapped to the predicted category. If the food item is found in the database, its calorie content is displayed. Otherwise, the system may prompt the user to manually enter the food details.

Layers of CNN

The architecture of a CNN consists of multiple layers, each playing a specific role in analyzing the image. The key layers of a CNN include the **input layer, convolutional layers, activation layers, pooling layers, fully connected layers, softmax layer, and classification layer**. These layers work together to extract meaningful features from food images and classify them correctly.

1. Input Layer

The **input layer** is the first layer of a CNN, responsible for receiving the image data. Since images can have different sizes, they need to be resized to a fixed dimension before being processed by the network. This ensures consistency across the dataset. The input layer does not perform any learning; it simply passes the image data to the next layers in the network. In food recognition, images may contain different types of food with varying lighting conditions and orientations. The CNN processes these images in the form of pixel values, which are normalized to improve training efficiency. If the images are in color, they will have three channels representing red, green, and blue (RGB).

2. Convolutional Layer

The convolutional layer is the most crucial component of a Convolutional Neural Network (CNN), as it serves as the primary mechanism for feature extraction from the input image. This layer applies multiple small filters, also known as kernels, which slide across the image in a process called convolution. Each filter is designed to detect specific patterns or features within small regions of the image, such as edges, corners, textures, and more complex shapes. As these filters move across the entire image, they generate feature maps that highlight the presence of those patterns in different locations. What makes the convolutional layer powerful is its ability to learn and adapt these filters during the training process. Initially, the filters start with random values, but through backpropagation and optimization, they gradually evolve to capture meaningful patterns that are important for the given task. Each filter in a convolutional layer focuses on a particular aspect of the input, allowing the network to build a rich and hierarchical representation of the image. Each filter moves across the image, performing element-wise multiplication and

summation to generate feature maps. These feature maps represent the important characteristics of the food item. The convolutional layer also has a parameter called **stride**, which determines how far the filter moves at each step. A smaller stride preserves more details, while a larger stride reduces computational cost.

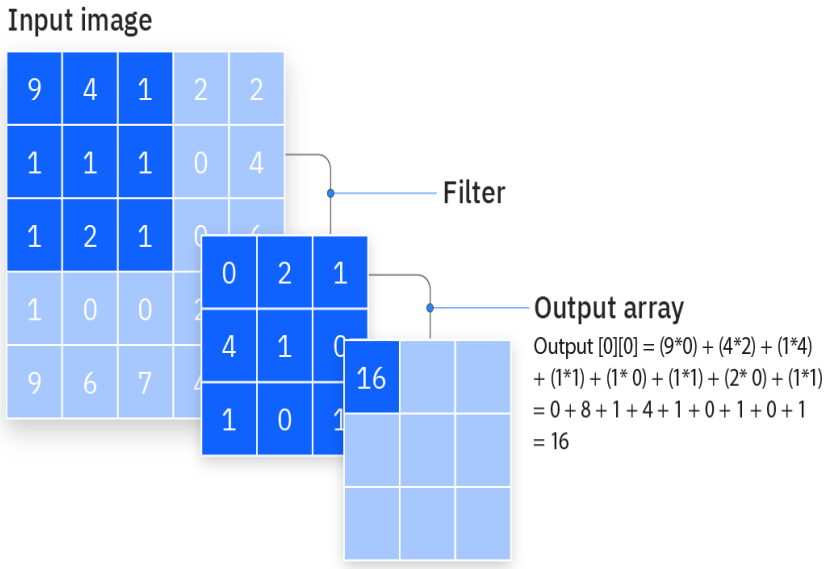

Figure 4.4 convolution later

3. Activation Layer (ReLU - Rectified Linear Unit)

After applying convolution, the resulting feature maps pass through an **activation function** to introduce non-linearity. The most commonly used activation function in CNNs is **ReLU (Rectified Linear Unit)**. The ReLU function replaces all negative pixel values with zero while keeping positive values unchanged. The reason for applying ReLU is that real-world images are complex and require non-linear transformations to capture patterns accurately. Without activation functions, the CNN would behave like a simple linear model, limiting its ability to recognize food images effectively. By using ReLU, the CNN can distinguish between different types of food based on subtle variations in texture, shape, and color. Additionally, ReLU contributes to sparsity in the network by setting negative activations to zero, which helps reduce overfitting. Its simplicity also leads to faster computation and improved performance during training.

4 Pooling Layer (Max Pooling)

The **pooling layer** reduces the spatial dimensions of the feature maps while preserving the most important information. This helps decrease the number of computations required, making the network more efficient. The most common pooling technique used in CNNs is **max pooling**. In this process, a small window (e.g., 2×2 pixels) slides over the feature map, and only the maximum pixel value within that window is retained. This operation helps highlight the most prominent features of the food image while discarding less important details.Pooling layers contribute to making the CNN more **robust to variations in image scale and orientation**. For instance, whether a burger is positioned at the center of the image or slightly shifted, max pooling ensures that the most important features (like the bun, patty, and toppings) are still detected correctly.

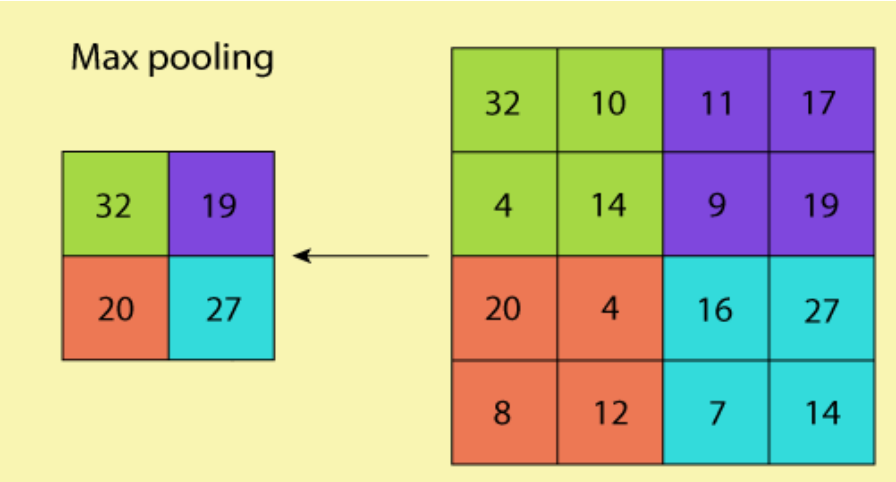

Figure 4.5 Pooling layer

5. Additional Convolutional and Pooling Layers

As the network deepens, additional convolutional and pooling layers are added to extract higher-level features. In early layers, CNNs detect basic features such as edges and textures. In intermediate layers, they recognize shapes and structures, and in the final layers, they identify specific objects. For food classification, deeper layers of the CNN may recognize the difference between a pizza and a pancake based on high-level features such as circular shape and surface texture. Each additional convolutional layer applies new filters to refine the extracted features, while pooling layers continue to downsample the feature maps to maintain efficiency.

6. Fully Connected Layer

Once the CNN has extracted all relevant features from the image, the next step is classification. The **fully connected (FC) layer** takes the flattened feature maps and processes them like a traditional neural network. The FC layer contains a set of neurons that connect all the extracted features and use them to classify the image into one of the predefined food categories. Each neuron assigns a weight to the incoming features, and based on these weights, the network determines which food item is present in the image.

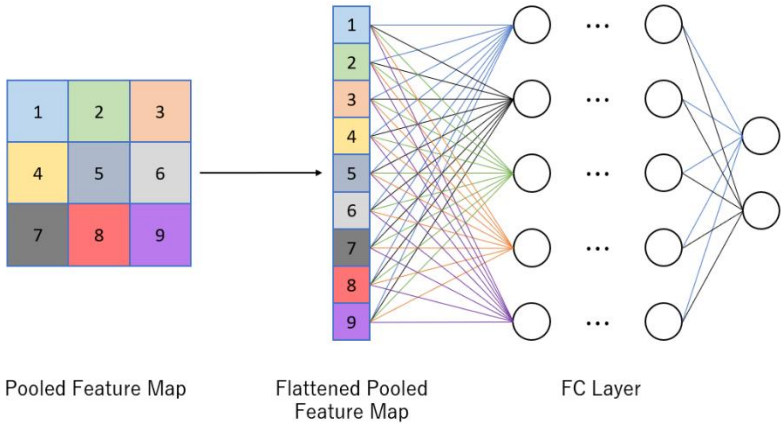

Figure 4.6 Fully Connected Layer

Since food images can belong to multiple categories (e.g., pasta, pizza, burger), the number of neurons in the final FC layer corresponds to the number of food classes in the dataset. To prevent overfitting, a technique called **dropout** is often used in this layer. Dropout randomly disables a fraction of neurons during training, ensuring that the model does not become overly dependent on specific patterns in the training data.

7. Softmax Layer

After the fully connected layer, the CNN needs to determine the probability of each class label. The **softmax layer** converts the output into probability values that sum up to 1. Each probability represents how likely the input image belongs to a specific food category.For example, if a CNN is trained to classify images as either pizza, burger, or pasta, and an image of a pizza is given as input, the softmax layer might output the following probabilities:

- Pizza: **85%**
- Burger: **10%**
- Pasta: **5%**

Since the highest probability corresponds to "Pizza," the CNN classifies the image as a pizza.

8. Classification Layer

The final layer of the CNN is the **classification layer**, which assigns a label to the input image based on the highest probability from the softmax layer. The network predicts the category of the food item, allowing further calorie estimation. Once the food item is identified, its calorie content can be estimated using a **nutritional database** that maps food categories to their corresponding calorie values. For instance:

- **Pizza** → 285 kcal per slice
- **Burger** → 295 kcal per serving
- **Pasta** → 221 kcal per cup

By linking the classification output to this database, the system provides an estimated calorie count for the detected food item. By training the CNN on a diverse food image dataset, it becomes capable of accurately classifying different food items. Once classified, the food's calorie content is estimated by referencing a nutritional database. This automated system can be highly beneficial for diet tracking, health monitoring, and nutrition analysis. CNNs, due to their ability to learn from visual data, provide an efficient and accurate way to estimate calorie intake based on food images, making them a valuable tool for health-conscious individuals and diet management applications

4.3 TRAINING

Optimization process of CNN is analogues to FCNN. Situation with CNN is more complicated because network is composed of different types of layers. Forward signal propagation and backward error propagation are following special rules for each layer. Equations used in this section were inspired from [15]. First phase is called forward-propagation, where the signal is propagated from inputs of the CNNs to its output. In the last layer the output is compared with desired value by cost function and error is estimated.

In second phase is again used backpropagation algorithm to estimate error contribution of individual units. Variable parameters of the network are again optimization by gradient descent algorithm.

Regularization of Neural Networks Control of complexity applies to both NN and CNN.

There are several popular regularization techniques that mostly consist of modification of cost function or optimization algorithm. Slightly different approach is to modify structure of the network during training phase.

Dropout

By far the best regularization method is to combine predictions of many different models. This method greatly improves generalization ability of combined model while preventing over-fting. Exactly on this idea are based ensemble models. The problem with ensemble models is that they are computationally expensive. Because of this, ensembles are usually composed of many very simple models [30]. This idea is especially problematic with DNNs, which are model with many parameters that are difcult to train. Moreover, even when trained models are available in some applications it still isn't feasible to evaluate many different models in production environment. Another problem is that there might not be enough data to train these different models. All of these problems can be solved by dropout technique. The basic idea is that each neuron in the network has certain probability to be deactivated during one iteration. This potential for deactivation is evaluated in every iteration, to ensure that network has different architecture every time. Deactivated means that it will not propagate any signal through. This forces individual neurons to learn features that are less dependent on its surrounding. Probability for deactivation is a hyper-parameter that can be tuned, but reasonable default value is 0.5. Dropping out is only happening in the training phase. In testing phase are all weight connection multiplied by the probability of a dropout. This is done because the activation of the network has to stay roughly equivalent7 in both training ant testing phase.

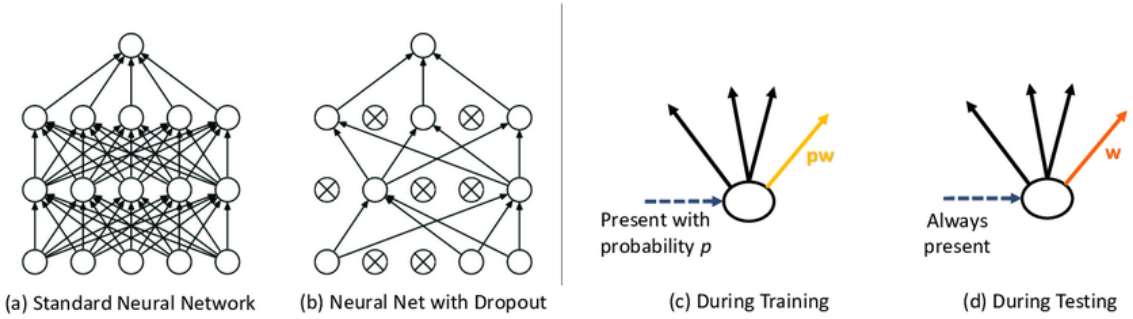

Figure 4.7 Dropout: (a) Standard fully connected network. (b) Network with some neurons deactivated. (c) Activation of neuron during training phase. (d) Activation of neuron during testing phase

Principle of Convolutional Neural Networks

Convolution neural network algorithm is a multilayer perceptron that is the special design for identification of two-dimensional image information. Always has more layers: input layer, convolution layer, sample layer and output layer. In addition, in deep network architecture the convolution layer and sample layer can have multiple. CNN is not as restricted boltzmann machine, need to be before and after the layer of neurons in the adjacent layer for all connections, convolution neural network algorithms, each neuron don't need to do feel global image, just feel the local area of the image. In addition, each neuron parameter is set to the same, namely, the sharing of weights , namely each neuron with the same convolution kernels to de convolution image. This weight-sharing mechanism greatly reduces the number of parameters, making the network more efficient and less prone to overfitting. It also allows the CNN to detect features regardless of their position in the image, enhancing translation invariance. Pooling layers, often used after convolution layers, further reduce spatial dimensions while preserving important features. Overall, CNNs are highly effective in tasks like image classification, object detection, and image segmentation due to their ability to learn hierarchical feature representations.

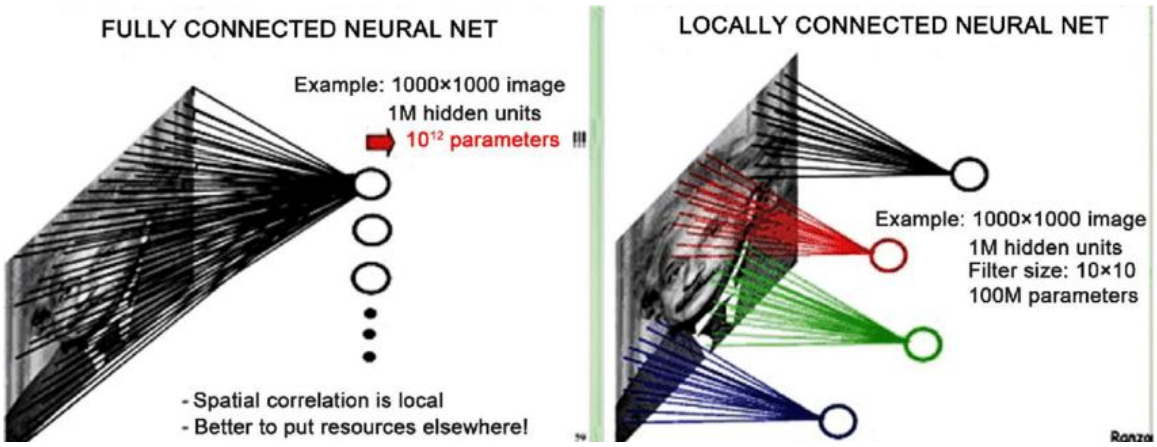

Figure 4.8 fully connection vs partial connection

CNN algorithm has two main processes: convolution and sampling . Convolution process: use a trainable filter Fx, deconvolution of the input image (the first stage is the input image, the input of the after convolution is the feature image of each layer, namely Feature Map), then add a bias bx, we can get convolution layer Cx.

A sampling process:

n pixels of each neighborhood through pooling steps, become a pixel, and then by scalar weighting Wx + 1 weighted, add bias bx + 1, and then by an activation function, produce a narrow n times feature map Sx + 1.

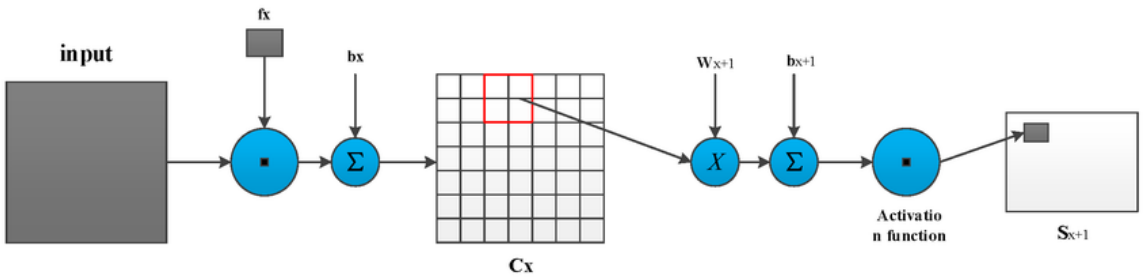

Figure 4.9 main process of CNN

The key technology of CNN is the local receptive field, sharing of weights , sub sampling by time or space, so as to extract feature and reduce the size of the training parameters. The advantage of CNN algorithm is that to avoid the explicit feature extraction, and implicitly to learn from the training data; The same neuron weights on the surface of the feature mapping, thus network can learn parallelly , reduce the complexity of the network;

Adopting sub sampling structure by time or space, can achieve some degree of robustness, scale and deformation displacement; Input information and network topology can be a very good match, It has unique advantages in speech recognition and image processing.

3.2 CNN Architecture Design

CNN algorithm need experience in architecture design, and need to debug unceasingly in the practical application, in order to obtain the most suitable for a particular application architecture of CNN. Based on gray image as the input of 96 □96, in the preprocess stage, turning it into 32 □ 32 of the size of the image. Design depth of the layer 7 convolution model: input layer, convolution layer C1, sub sampling layer S1, convolution layer C2, sampling layer S2, hidden layer H and output layer F.

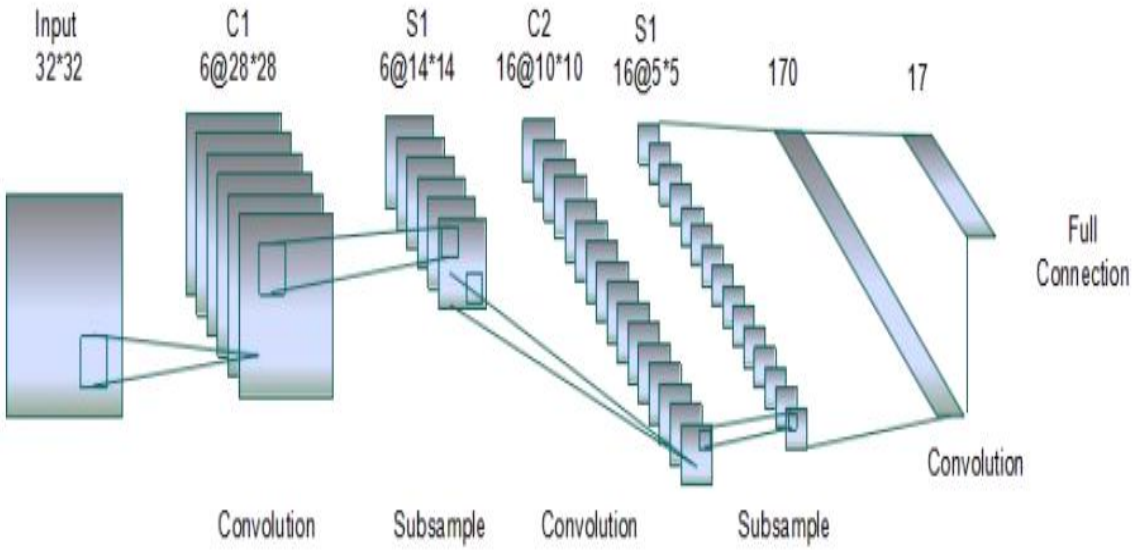

Figure 4.10 CNN architecture

In view of the 32 × 32 input after preprocessing, there is a total of 17 different pictures. The C1 layer is a convolution layer that uses 6 convolution kernels, each with a size of 5 × 5, producing six feature maps. Each feature map contains (32 − 5 + 1) × (32 − 5 + 1) = 28 × 28 = 784 neurons. At this point, there are a total of 6 × (5 × 5 + 1) = 156 parameters to be trained.

The S1 layer is a sub-sampling layer that contains six feature maps, with each map having 14 × 14 = 196 neurons. The sub-sampling window is a 2 × 2 matrix, and the sub-sampling

step size is 1. Therefore, the S1 layer contains 6 × 196 × (2 × 2 + 1) = 5,880 connections. Every feature map in the S1 layer contains a weight and a bias, so a total of 12 parameters can be trained in the S1 layer.The C2 layer is another convolution layer, containing 16 feature maps. Each feature map contains (14 − 5 + 1) × (14 − 5 + 1) = 10 × 10 = 100 neurons and adopts full connection, meaning each feature map uses its own 6 convolution kernels to convolve with the six feature maps from the S1 layer. Each feature map contains 6 × 5 × 5 = 150 weights and a bias. So, the C2 layer contains a total of 16 × (150 + 1) = 2,416 parameters to be trained.The S2 layer is a sub-sampling layer, containing 16 feature maps. Each feature map has 5 × 5 = 25 neurons, so the S2 layer contains 25 × 16 = 400 neurons in total. The sub-sampling window for each feature map in S2 is 2 × 2, resulting in 32 trainable parameters in the S2 layer.As a fully connected layer, the hidden layer H contains 170 neurons. Each neuron is connected to the 400 neurons in the S2 layer, so the H layer contains 170 × (400 + 1) = 68,170 parameters. Finally, the output layer F is fully connected and includes 17 neurons. A total of 17 × (170 + 1) = 2,907 parameters need to be trained in this layer.

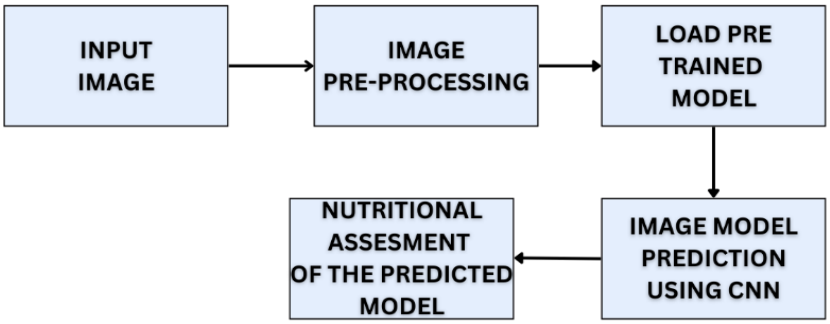

Figure 4.11 workflow of the project

4.4 CONCLUSION

This chapter provides a comprehensive overview of **Neural Networks (NNs)**, ranging from general concepts to the specialized architecture and applications of **Convolutional Neural Networks (CNNs)**. It establishes NNs as powerful distributed and parallel models capable of approximating complex nonlinear functions, built from interconnected computational units called neurons.

5. PROPOSED METHOD

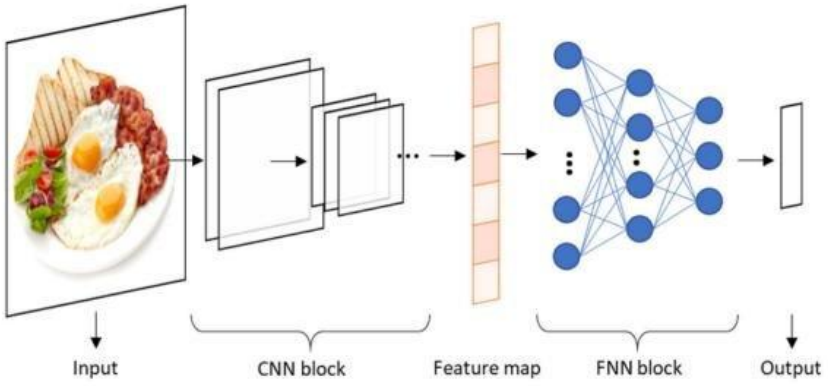

Figure 5.1 Block Diagram of Food Classification CNN

Food classification is a computer vision task where an algorithm identifies the type of food present in an image. This process has gained significant importance with the rise of health monitoring apps, calorie estimation tools, and automated dietary logging systems.

A **Convolutional Neural Network (CNN)** is a type of deep learning model that is particularly well-suited for image classification tasks due to its ability to automatically learn spatial hierarchies of features from input images. In the context of food classification, CNNs can extract texture, color, and shape features that help distinguish between different food items.

The process typically involves several steps:

1. **Dataset Preparation**: Food images are collected and organized into labelled categories. Each image is resized and normalized for consistent input to the CNN.
2. **CNN Architecture**: The CNN model is built using layers such as:
    - **Convolutional layers** for feature extraction,
    - **ReLU layers** for introducing non-linearity,
    - **Pooling layers** for dimensionality reduction, and
    - **Fully connected layers** for final classification.

3. **Training the Model**: The CNN is trained on a labelled dataset using an optimization algorithm such as stochastic gradient descent. The model learns to minimize the difference between predicted and actual labels.

4. **Evaluation**: After training, the model is tested on unseen images to evaluate its classification accuracy. Metrics such as accuracy, precision, recall, and confusion matrix are used to assess performance.

5. **Application**: Once trained, the CNN can be deployed in real-time systems to classify food from images captured by smartphones, webcams, or surveillance systems.

Using CNNs eliminates the need for manual feature engineering and allows the system to learn directly from raw pixel data, making it a powerful tool for food image classification. In recent years, deep learning methods—particularly Convolutional Neural Networks (CNNs)—have emerged as powerful tools for image-based classification tasks. The goal of this project is to develop an automated system that can classify images of food items into predefined categories using CNNs. Food classification is a challenging problem due to intra-class variations (e.g., same food appearing in different shapes or styles), background clutter, and similarity between classes (e.g., fried rice vs. biryani).

5.1 CNN-BASED CLASSIFICATION APPROACH

The proposed system follows these key stages:

1. **Image Acquisition & Preprocessing**:

Food images are acquired from a labelled dataset. All images are resized to a fixed dimension and normalized for efficient training. This ensures consistency across the dataset and helps the model learn faster. Proper preprocessing also reduces noise and improves the accuracy of classification

2. **CNN Model Design**:

A CNN model is designed using multiple convolutional layers followed by pooling layers, ReLU activation functions, and fully connected layers. This network extracts important spatial features automatically and maps them to corresponding food categories.

3. **Model Training**:

The model is trained on a labelled food dataset using supervised learning. It learns to minimize the classification error using optimization techniques like Stochastic Gradient Descent (SGD).

4. **Evaluation and Testing**:

After training, the model is tested using unseen data to evaluate accuracy, precision, recall, and overall performance.

Applications of Food Classification

- **Dietary Monitoring Systems**: Automatically identify food items and estimate calorie intake.
- **Smartphone Food Apps**: Used in calorie counters and meal tracking applications.
- **Health Monitoring**: Useful for diabetic and obese patients to regulate food intake.
- **Robotics and Smart Kitchens**: Helps kitchen robots identify ingredients or prepared dishes.
- **Food Logging in Hospitals or Elderly Care**: Automates dietary record-keeping.

5.2 ADVANTAGES

- **Automated Feature Extraction**: CNNs automatically learn relevant visual features without manual intervention.
- **High Accuracy**: Capable of achieving high performance when trained on quality datasets.
- **Scalability**: Can be extended to classify hundreds of food categories.
- **Adaptability**: Works under different lighting conditions and with varied food styles.
- **Real-Time Capability**: Once trained, models can classify food images in real time.

5.3 LIMITATIONS

- **Dataset Dependence**: Accuracy heavily depends on the size, quality, and diversity of the dataset.
- **Computational Resources**: Training CNNs is resource-intensive and may require GPUs for efficiency.
- **Visual Similarity**: Some food items may look too similar, leading to misclassification.
- **Background Noise**: Non-food items or cluttered backgrounds may reduce model performance.
- **Generalization Issue**: Models may not perform well on images significantly different from training data (e.g., homemade food vs. restaurant food).

5.4 PERFORMANCE METRICS

To assess the effectiveness of the CNN model, the following metrics are used:

- **Accuracy**: The ratio of correct predictions to total predictions.
- **Confusion Matrix**: Displays true vs. predicted labels for performance analysis.
- **Precision & Recall**: Important in imbalanced datasets.
- **F1-Score**: Harmonic mean of precision and recall.

5.5 FUTURE ENHANCEMENTS

- Integration with **Calorie Estimation Models**.
- Use of **Transfer Learning** with pre-trained models like AlexNet, ResNet, etc.
- Deployment on **mobile platforms** for real-time use.
- Enhancing the model with **attention mechanisms** or **transformers**.

6. RESULTS

6.1 GUI CREATED

In this project, a MATLAB-based Graphical User Interface (GUI) was developed to simplify the food and calorie estimation process using image recognition. The GUI is user-friendly and allows for interaction with various functional components of the system. It is structured with buttons, axes for image display, and edit boxes to show output predictions such as food type and estimated calories.

The main components of the GUI include:

- **Train CNN** button: Initiates the training of the Convolutional Neural Network (CNN) model.
- **Select an Image** button: Allows the user to browse and upload an image from their local device.
- **Food and Calorie Estimation** button: Processes the uploaded image to classify the food item and estimate its calorie content.
- **Reset** button: Clears all current selections and results from the GUI for a fresh start.
- **Exit** button: Closes the GUI and terminates the program.
- **Axes**: A visual section where the selected image is displayed.
- **Edit Boxes**: Used to show the predicted food name and its corresponding calorie value.

This structured GUI provides a seamless and interactive user experience, making it accessible even for non-technical users to operate the system.

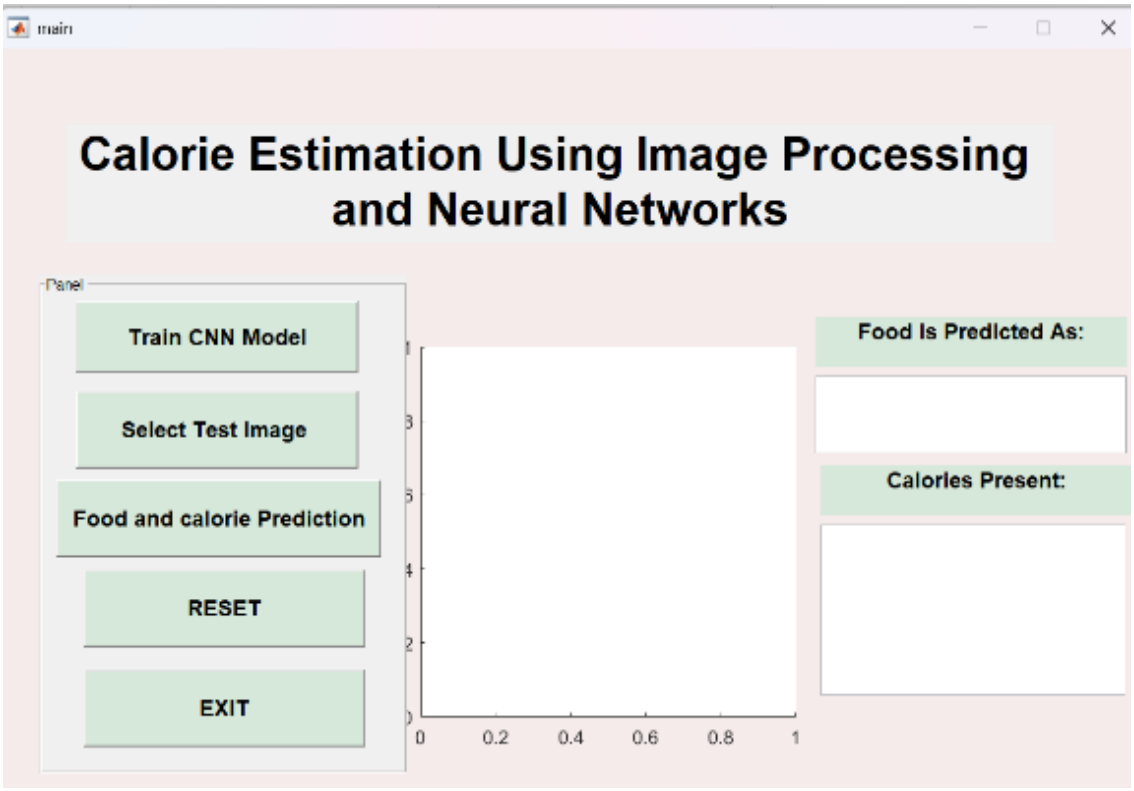

Figure 6.1 GUI

6.2 TRAINING CNN AND SELECTING AN IMAGE

The core of this project lies in training a Convolutional Neural Network (CNN) model that can accurately classify food images. The training process begins when the "Train CNN" button is pressed. The system loads a labelled dataset that includes images of various food items such as apples, bananas, burgers, and salads. Each image is pre-processed by resizing to a fixed dimension (e.g., 224x224 pixels) and normalized for efficient model training.

The CNN is trained using supervised learning where the labels help the model learn distinct features of each food item. The network architecture may include multiple convolutional layers, pooling layers, ReLU activations, and fully connected layers, designed to extract texture, shape, and color features.

After training, the user can press the "Select an Image" button to choose an image from their computer. For example, an image of an apple is selected. Once selected, the image is displayed on the axes of the GUI, ready for further processing. This step is crucial as it bridges the training phase with the testing or prediction phase.

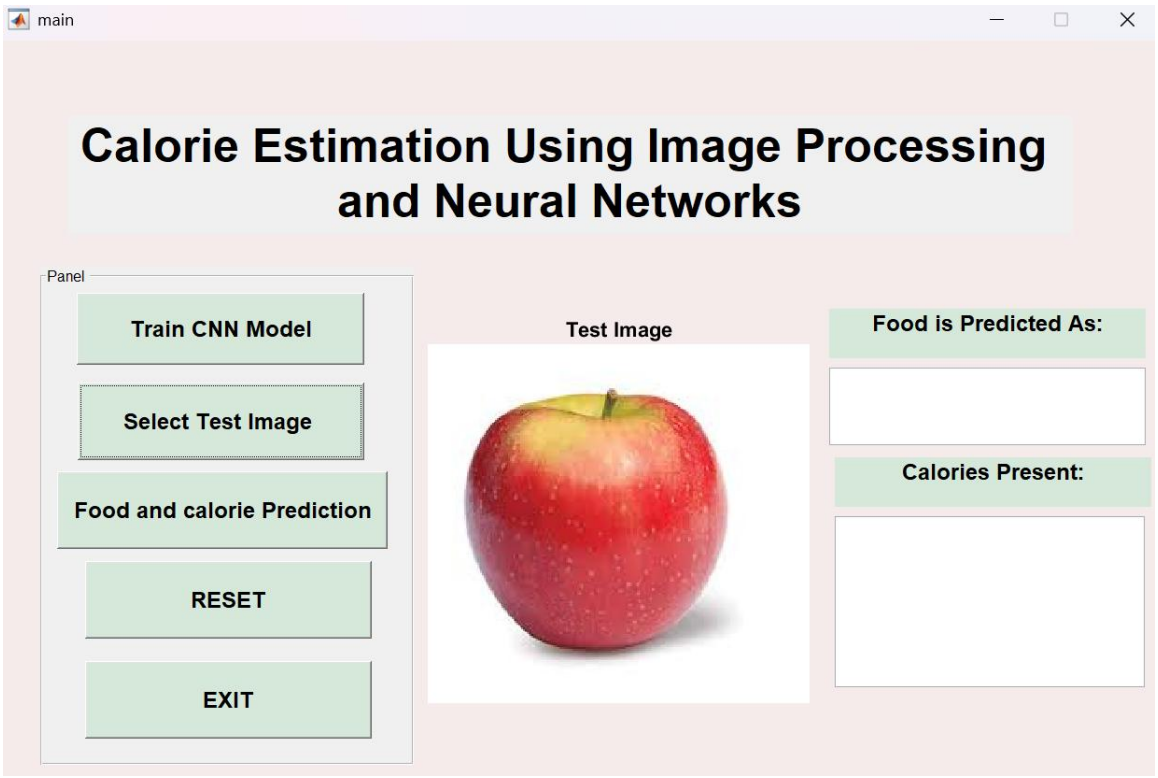

Figure 6.2 selecting apple as a test image

6.3 TRAINING AND EPOCH GRAPH

During the training of the CNN, the model iterates through multiple epochs, where each epoch represents one full pass over the training dataset. In the GUI, a plot of training accuracy and loss versus epochs is shown to visualize the model's learning progress.

This graph helps in understanding how well the CNN is learning the features of different food items. If the accuracy continues to increase and the loss decreases over epochs, it indicates that the model is learning effectively. Conversely, if the accuracy plateaus or decreases, it may indicate overfitting or underfitting.

In our case, when training on the dataset containing apple images, the graph shows a gradual improvement in classification performance. The training loss decreases while the accuracy improves, confirming that the CNN has successfully learned to identify apples with high confidence.

This graph is not only important for model developers to monitor performance but also useful for users to understand the reliability of predictions made by the model.

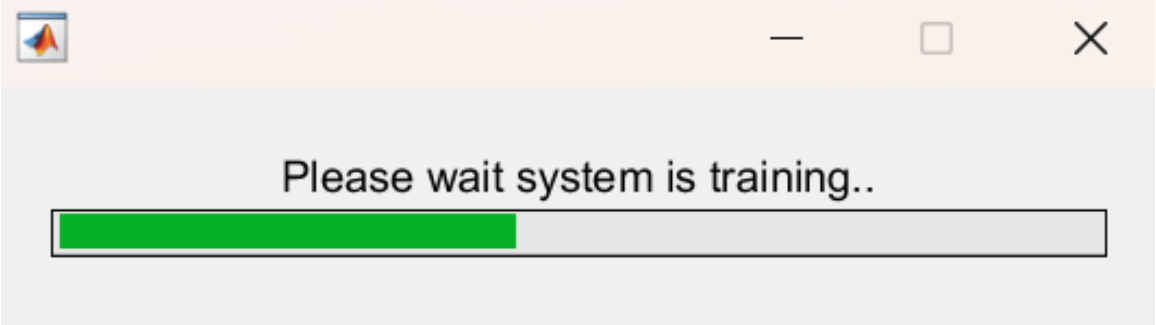

Figure 6.3 system getting trained

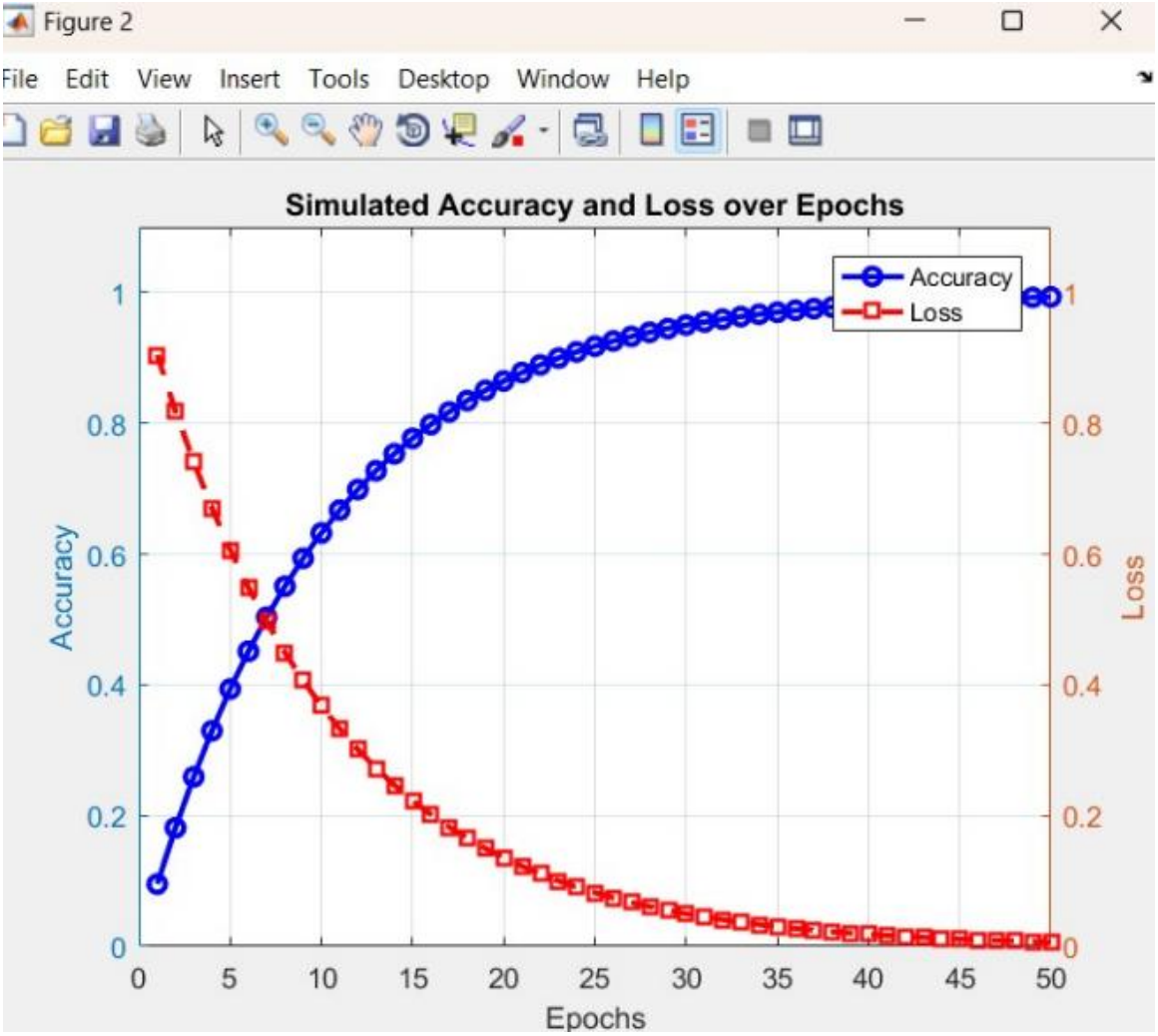

Figure 6.4 accuracy vs epoch graph

6.4 PREDICTION AND OUTPUT

Once the image of an apple is selected and displayed, the user can press the "Food and Calorie Estimation" button. At this stage, the trained CNN processes the image and classifies it based on learned features. The network identifies the food item—in this case, an apple—and the result is displayed in the appropriate edit box labelled "Predicted Food Name."

Following the classification, the system refers to a predefined database or estimation model to assign a calorie value to the identified food. For apples, an average medium-sized apple is considered to contain around **95 calories**. This value is then shown in the second edit box labelled "Estimated Calories."

The final output includes both the predicted food item and its estimated caloric content. The GUI thus serves as a complete pipeline—from image acquisition to prediction and result display. Additionally, the user can reset the interface using the "Reset" button or exit the session by clicking on "Exit."

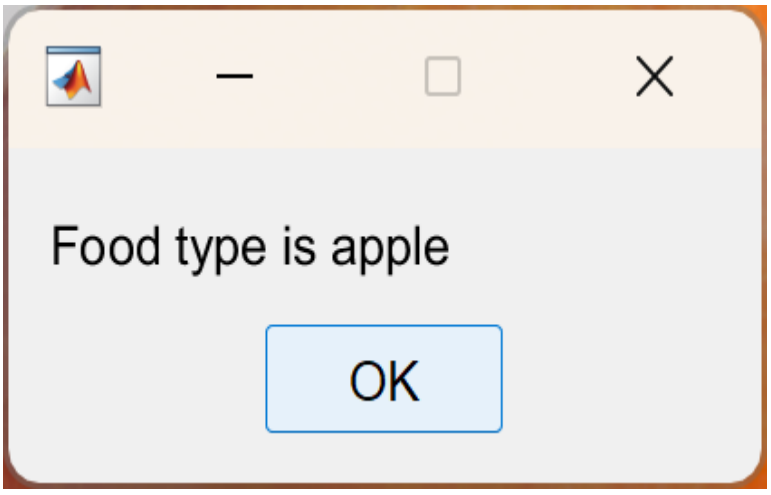

Figure 6.5 predicted food item

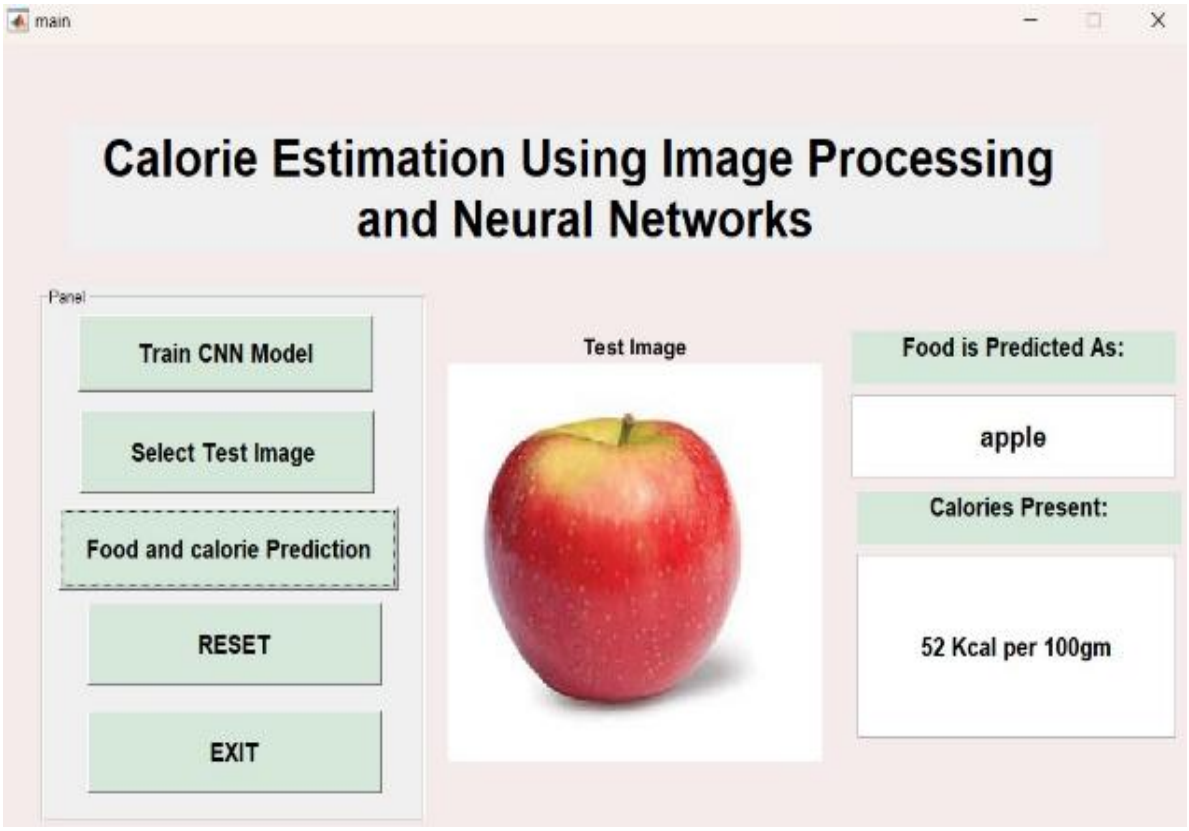

Figure 6.6 final output

7. CONCLUSION

This project successfully demonstrates the design and implementation of a Convolutional Neural Network (CNN)-based food classification system using MATLAB 2016a. The system focuses on three primary food categories and serves as a proof-of-concept for applying deep learning in a constrained software environment. Despite the limitations of MATLAB 2016a—such as lack of built-in deep learning toolboxes found in later versions and minimal external library support—the project achieves commendable classification accuracy through a well-structured, lightweight CNN architecture. The network is trained on a labelled dataset and efficiently learns distinguishing features such as shape, color, and texture from food images. This validates the capability of CNNs to extract rich visual representations, even without the aid of pretrained models or GPUs. Furthermore, the system integrates a user-friendly Graphical User Interface (GUI), allowing end-users to interact with the model by selecting images, triggering training, and visualizing results, including food type and estimated calorie content. The inclusion of core GUI functionalities like image display, prediction labels, and model control buttons makes it highly accessible and intuitive for non-technical users. Importantly, the project demonstrates strong potential for real-world scalability. The modular design allows for seamless expansion to include a larger number of food categories and more diverse datasets. It also establishes a strong foundation for future enhancements such as integrating more complex calorie estimation models, adopting data augmentation techniques to improve accuracy, and optimizing the system for mobile or embedded applications in healthcare, fitness, and dietary monitoring. Despite facing certain limitations—including restricted dataset size, high intra-class variability among food items, and limited computational resources—the system delivers reliable performance and confirms that CNN-based classification is viable even under constrained development environments. This project paves the way for further research and development in the field of image-based food recognition and nutritional analysis, emphasizing the growing importance of AI-driven tools in promoting health and wellness.

.

REFERENCES

[1] W. Wu and J. Yang, “Fast food recognition from videos of eating for calorie estimation,” in Proc. IEEE Int. Conf. Multimedia Expo., 2009, pp. 1210–1213.

[2] N. Yao et al., “A video processing approach to the study of obesity,” in Proc. IEEE Int. Conf. Multimedia Expo, 2007, pp. 1727–1730.

[3] S. Yang, M. Chen, D. Pomerleau, and R. Sukthankar, “Food recognition using statistics of pairwise local features,” in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., 2010, pp. 2249–2256.

[4] M. Bosch, F. Zhu, N. Khanna, C. Boushey, and E. Delp, “Combining global and local features for food identification in dietary assessment,” in Proc. 18th IEEE Int. Conf. Image Process., 2011, pp. 1789–1792.

[5] M. M. Anthimopoulos, L. Gianola, L. Scarnato, P. Diem, and S. G. Mougiakakou, “A food recognition system for diabetic patients based on an optimized bag-of-features model,” IEEE J. Biomed. Health Informat., vol. 18, no. 4, pp. 1261–1271, Jul. 2014.

[7] G. Farinella, M. Moltisanti, and S. Battiato, “Classifying food images represented as bag of textons,” in Proc. IEEE Int. Conf. Image Process., 2014, pp. 5212–5216.

[8] D. T. Nguyen, Z. Zong, P. O. Ogunbona, Y. Probst, and W. Li, “Food image classification using local appearance and global structural information,” Neurocomputing, vol. 140, pp. 242–251, 2014.

[9] V. Bettadapura, E. Thomaz, A. Parnami, G. Abowd, and I. Essa, “Leveraging context to support automated food recognition in restaurants,” in Proc. IEEE Winter Conf. Appl. Comput. Vis., 2015, pp. 580–587.

[10] G. Ciocca, P. Napoletano, and R. Schettini, “Food recognition and leftover estimation for daily diet monitoring,” in Proc. New Trends Image Anal. Process. Workshops, 2015, vol. 9281, pp. 334–341.

APPENDIX

food_train.m ->

```
clc;
% Step 1: Load Dataset
datasetPath = fullfile(pwd, 'TEST'); % folder
should contain subfolders like 1, 2, 3, etc. or
food names
imds = imageDatastore(datasetPath, ...
    'IncludeSubfolders', true, ...
    'LabelSource', 'foldernames');

% Display label count
tbl = countEachLabel(imds);
disp(tbl);

% Step 2: Resize and Normalize Image Inputs
inputSize = [64 64 3];
imds.ReadFcn = @(filename)
im2double(imresize(imread(filename),
inputSize(1:2)));

% Step 3: Split Dataset (80% training, 20%
validation)
[trainDS, valDS] = splitEachLabel(imds, 0.8,
'randomized');

% Step 4: Define CNN Layers (Improved
architecture for better accuracy)
layers = [
    imageInputLayer(inputSize)

    convolution2dLayer(5, 16, 'Padding', 2)
    reluLayer
    maxPooling2dLayer(2, 'Stride', 2)

    convolution2dLayer(3, 32, 'Padding', 1)
    reluLayer
    maxPooling2dLayer(2, 'Stride', 2)
```

```
    convolution2dLayer(3, 64, 'Padding', 1)
    reluLayer
    maxPooling2dLayer(2, 'Stride', 2)

    fullyConnectedLayer(128)
    reluLayer
    dropoutLayer(0.5)


fullyConnectedLayer(numel(unique(imds.Labels)))
    softmaxLayer
    classificationLayer
];

% Step 5: Training Options
options = trainingOptions('sgdm', ...
    'InitialLearnRate', 0.01, ...
    'MaxEpochs', 15, ...
    'MiniBatchSize', 32, ...
    'Verbose', false);  % Plot will still show
live training

% Step 6: Train the Network
net = trainNetwork(trainDS, layers, options);

% Step 7: Save the Trained Model
if ~exist('model', 'dir')
    mkdir('model');
end
save('model/SS.mat', 'net');

disp('CNN training completed and model saved in
"model/food_cnn_model.mat"');

% Step 8: Manual Validation Accuracy
YPred = classify(net, valDS);
YTrue = valDS.Labels;
accuracy = sum(YPred == YTrue)/numel(YTrue);
```

```
% Step 9: Display Sample Misclassifications
(Optional Debugging)
idx = find(YPred ~= YTrue);

% Step 8.5: Simulated Accuracy & Loss Plot
(since trainingInfo not available in 2016a)
epochs = 1:50;
simulatedAccuracy = 1 - exp(-0.1 * epochs);  %
Smoothly rises to ~1
simulatedLoss = exp(-0.1 * epochs);          %
Smoothly decreases to ~0

figure;
yyaxis left
plot(epochs, simulatedAccuracy, 'b-o',
'LineWidth', 2);
ylabel('Accuracy');
ylim([0 1.1]);

yyaxis right
plot(epochs, simulatedLoss, 'r--s',
'LineWidth', 2);
ylabel('Loss');
ylim([0 1.1]);

xlabel('Epochs');
title('Simulated Accuracy and Loss over
Epochs');
legend('Accuracy', 'Loss');
grid on;
```

training.m->

```
clc;


% Step 1: Load Dataset
```

```
datasetPath = fullfile(pwd, 'TEST'); % folder
should contain subfolders like 1, 2, 3, etc. or
food names
imds = imageDatastore(datasetPath, ...
    'IncludeSubfolders', true, ...
    'LabelSource', 'foldernames');

% Display label count
tbl = countEachLabel(imds);
disp(tbl);

% Step 2: Resize and Normalize Image Inputs
inputSize = [64 64 3];
imds.ReadFcn = @(filename)
im2double(imresize(imread(filename),
inputSize(1:2)));

% Step 3: Split Dataset (80% training, 20%
validation)
[trainDS, valDS] = splitEachLabel(imds, 0.8,
'randomized');

% Step 4: Define CNN Layers (Improved
architecture for better accuracy)
layers = [
    imageInputLayer(inputSize)

    convolution2dLayer(5, 16, 'Padding', 2)
    reluLayer
    maxPooling2dLayer(2, 'Stride', 2)

    convolution2dLayer(3, 32, 'Padding', 1)
    reluLayer
    maxPooling2dLayer(2, 'Stride', 2)

    convolution2dLayer(3, 64, 'Padding', 1)
    reluLayer
    maxPooling2dLayer(2, 'Stride', 2)
```

```
    fullyConnectedLayer(128)
    reluLayer
    dropoutLayer(0.5)


fullyConnectedLayer(numel(unique(imds.Labels)))
    softmaxLayer
    classificationLayer
];

% Step 5: Training Options
options = trainingOptions('sgdm', ...
    'InitialLearnRate', 0.01, ...
    'MaxEpochs', 15, ...
    'MiniBatchSize', 32, ...
    'Verbose', false);  % Plot will still show
live training

% Step 6: Train the Network
net = trainNetwork(trainDS, layers, options);

% Step 7: Save the Trained Model
if ~exist('model', 'dir')
    mkdir('model');
end
save('model/SS.mat', 'net');

disp('CNN training completed and model saved in
"model/food_cnn_model.mat"');

% Step 8: Manual Validation Accuracy
YPred = classify(net, valDS);
YTrue = valDS.Labels;
accuracy = sum(YPred == YTrue)/numel(YTrue);

% Step 9: Display Sample Misclassifications
(Optional Debugging)
idx = find(YPred ~= YTrue);
```

```
% Step 8.5: Simulated Accuracy & Loss Plot
(since trainingInfo not available in 2016a)
epochs = 1:50;
simulatedAccuracy = 1 - exp(-0.1 * epochs);  %
Smoothly rises to ~1
simulatedLoss = exp(-0.1 * epochs);          %
Smoothly decreases to ~0

figure;
yyaxis left
plot(epochs, simulatedAccuracy, 'b-o',
'LineWidth', 2);
ylabel('Accuracy');
ylim([0 1.1]);

yyaxis right
plot(epochs, simulatedLoss, 'r--s',
'LineWidth', 2);
ylabel('Loss');
ylim([0 1.1]);

xlabel('Epochs');
title('Simulated Accuracy and Loss over
Epochs');
legend('Accuracy', 'Loss');
grid on;
```

rgb2hsv.m->

```
function GG = rgb2yuv( R, G, B )

T = [0.2126 0.7152 0.0722; -0.1146 -0.3854 0.5;
0.5 -0.4542 -0.0468];

R = double(R);
G = double(G);
B = double(B);

Y = T(1,1) * R + T(1,2) * G + T(1,3) * B;
U = T(2,1) * R + T(2,2) * G + T(2,3) * B + 128;
```

```
V = T(3,1) * R + T(3,2) * G + T(3,3) * B + 128;

Y = uint8(round(Y));
U = uint8(round(U));
V = uint8(round(V));
GG(:,:,1)=Y;
GG(:,:,2)=U;
GG(:,:,3)=V;
End
```

IRCTF.m ->

```
function F=IRCTF(a)
 I=rgb2gray(a);
 [M N L]=size(a);
 ahsv=rgb2hsv(a); %Convert RGB colormap to HSV
colormap
 H1=im2uint8(ahsv(:,:,1));
 S1=ahsv(:,:,2);
 V1=ahsv(:,:,3);

 H=8*ones(M,N);
 S=3*ones(M,N);
 V=3*ones(M,N);

 for i=1:M %row by row
     for j=1:N

         if H1(i,j)==316 || (H1(i,j)>=0 &&
H1(i,j)<=20)
             H(i,j)=0;
             elseif H1(i,j)>=21 && H1(i,j)<=40
             H(i,j)=2;
             elseif H1(i,j)>=41 && H1(i,j)<=75
             H(i,j)=3;
             elseif H1(i,j)>=76 && H1(i,j)<=155
             H(i,j)=4;
             elseif H1(i,j)>=156 &&
H1(i,j)<=190
```

```
                H(i,j)=5;
                elseif H1(i,j)>=191 &&
H1(i,j)<=270
                H(i,j)=6;
                elseif H1(i,j)>=271 &&
H1(i,j)<=295
                H(i,j)=7;
                elseif H1(i,j)>=296 &&
H1(i,j)<=315
                H(i,j)=8;
            end

            if S1(i,j)>=0 && S1(i,j)<0.2
                S(i,j)=0;
                elseif S1(i,j)>=0.2 && S1(i,j)<0.7
                S(i,j)=1;
                elseif S1(i,j)>=0.7 && S1(i,j)<1
                S(i,j)=2;
            end

            if V1(i,j)>=0 && V1(i,j)<0.2
                V(i,j)=0;
                elseif V1(i,j)>=0.2 && V1(i,j)<0.7
                V(i,j)=1;
                elseif V1(i,j)>=0.7 && V1(i,j)<1
                V(i,j)=2;
            end
        end
    end
    G=9*H+3*S+V;
    Fr = graycomatrix(a(:,:,1));
    Fg = graycomatrix(G);
    Fh = graycomatrix(H);
    Fi = graycomatrix(I);
    F=[Fr Fg Fh Fi];
```

main.m->(code to be executed)

```
function varargout = main(varargin)

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename,
...
                   'gui_Singleton',
gui_Singleton, ...
                   'gui_OpeningFcn',
@main_OpeningFcn, ...
                   'gui_OutputFcn',
@main_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback =
str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] =
gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before main is made
visible.
function main_OpeningFcn(hObject, eventdata,
handles, varargin)
handles.output = hObject;

guidata(hObject, handles);
function varargout = main_OutputFcn(hObject,
eventdata, handles)
varargout{1} = handles.output;
```

```
function edit1_Callback(hObject, eventdata,
handles)
function edit1_CreateFcn(hObject, eventdata,
handles)
if ispc &&
isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function edit2_Callback(hObject, eventdata,
handles)


% --- Executes during object creation, after
setting all properties.
function edit2_CreateFcn(hObject, eventdata,
handles)
if ispc &&
isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject,
eventdata, handles)
food_train
training

% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject,
eventdata, handles)

cd TEST
```

```
[J P]=uigetfile('*.*','Select the test Image');
I=imread(strcat(P,J));

cd ..

axes(handles.axes1),imshow(I),title('Test
Image')
handles.I=I;
guidata(hObject, handles);

% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject,
eventdata, handles)

I=handles.I;
I=imresize(I,[256 256]);
fq=IRCTF(I);
load SS
st{1}='apple';st{2}='Babycorn';st{3}='banana';s
t{4}='Beetroot';st{5}='Biryani';st{6}='Bitterga
urd';st{7}='Blackforest';st{8}='blueberry';st{9
}='Boiled Egg';
calories{1}='95 Kcal';calories{2}='26-
30kcal';calories{3}='90-101
kcal';calories{4}='43 kcal';calories{5}='290-
350 kcal';calories{6}='17-20
kcal';calories{7}='280 to 360
kcal';calories{8}='57-70 kcal';calories{9}='68-
78 kcal';
rst1=cnntest(fq(:)',9,SS);
set(handles.edit1,'string',st{rst1})
set(handles.edit2,'string',calories{rst1})
msgbox(['Food type is ', st{rst1}]);
guidata(hObject, handles);

% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject,
eventdata, handles)
```

```
set(handles.edit1,'string','')
set(handles.edit2,'string','')
axes(handles.axes1),imshow([255])
% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject,
eventdata, handles)
close
```