FOOD IMAGE CLASSIFICATION

ABSTRACT

Food image classification has become increasingly important in today's digital age, especially with the rise of online food delivery services and health-conscious consumers. However, ensuring the authenticity of food images used by restaurants and food delivery platforms remains a challenge. One common issue is the use of misleading images to attract customers, which can lead to dissatisfaction and mistrust. For food delivery platforms and restaurants, it is infeasible to manually verify the authenticity of these images among thousands of uploads. If sufficient data is collected and made available, machine learning algorithms can be applied to solve this problem. In this work, popular supervised and unsupervised machine learning algorithms have been applied to classify food images in a highly diverse dataset. It was found that convolutional neural networks (CNNs), particularly when combined with transfer learning techniques such as RESNET18, can handle the complexity and variability of food image and give the best classification results.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF SCREENSHOTS

1. INTRODUCTION

1.1 PURPOSE:

The primary purpose of this project is to develop a robust and efficient food image classification system that can accurately classify images of different food items into their respective categories. This system holds significant potential for various applications, including dietary assessment, food logging, and automated food recognition systems. The project leverages advanced deep learning techniques, specifically convolutional neural networks (CNNs), to achieve high classification accuracy. Additionally, the use of transfer learning techniques, particularly with the RESNET18 model, enhances the model's ability to generalize and improves its performance on the food image dataset.

In today's digital age, the proliferation of social media platforms has led to an exponential increase in the number of images uploaded daily. Among these, food images constitute a significant portion, as users frequently share pictures of their meals on platforms like Instagram, Facebook, and Snapchat. This trend presents both opportunities and challenges for various industries, particularly the food and beverage sector. For restaurants, hotels, and food delivery services, the ability to classify food images accurately can enhance customer engagement and satisfaction [1]. By analysing food images shared by customers, these businesses can gain valuable insights into consumer preferences and trends. This information can be used to tailor marketing strategies, improve menu offerings, and enhance the overall dining experience.

Moreover, food image classification can play a crucial role in dietary assessment and health monitoring. Nutritionists and healthcare professionals can use this technology to analyse the nutritional content of meals, track dietary habits, and provide personalized recommendations to individuals. This can be particularly beneficial for people with specific dietary needs, such as those with allergies, diabetes, or other health conditions. Automated food recognition systems, such as those used in smart refrigerators and kitchen appliances, can also benefit from food image classification. These systems can automatically identify and categorize food items, providing users with valuable information about the contents of their refrigerator or pantry. This can help users make informed decisions about their meals, track their dietary habits, and manage their food inventory more effectively.

Deep learning, a subset of machine learning, has revolutionized the field of image classification. Convolutional neural networks (CNNs), a type of deep learning model, are particularly well-suited for image classification tasks due to their ability to automatically learn and extract features from raw image data. CNNs consist of multiple layers, including convolutional layers, pooling layers, and fully connected layers, which work together to identify and classify objects within images [10]. The success of CNNs in image classification tasks can be attributed to their ability to capture spatial hierarchies in images. Convolutional layers apply filters to the input image to detect features such as edges, textures, and shapes. Pooling layers reduce the dimensionality of the feature maps, making the model more computationally efficient and robust to variations in the input image. Fully connected layers then use these features to make the final classification decision.

While deep learning models have shown remarkable success in image classification tasks, training these models from scratch requires large amounts of labelled data and computational resources. Transfer learning offers a solution to this challenge by leveraging pre-trained models that have been trained on large-scale datasets, such as ImageNet. These pre-trained models can then be fine-tuned on smaller, task-specific datasets to achieve high classification accuracy with less data and computational resources. The RESNET18 model, developed by Google, is a popular choice for transfer learning in image classification tasks. RESNET18 is a deep convolutional neural network that uses a unique architecture called Inception modules. These modules allow the model to capture multi-scale features by applying multiple filters of different sizes to the input image. This enables the model to learn both fine-grained and coarse-grained features, making it highly effective for image classification tasks.

In this project, we utilize the RESNET18 model pre-trained on the ImageNet dataset and fine-tune it on a food image dataset to achieve high classification accuracy. The use of transfer learning with the RESNET18 model allows us to leverage the learned features from the large-scale ImageNet dataset, enhancing the model's ability to generalize and improving its performance on the food image dataset. The project involves several key steps, including data collection, data pre-processing, model selection, model training, and model evaluation.

Data collection involves gathering a dataset of food images for training and validation. The dataset used in this project is the Indian Food Image dataset, which contains 8 food categories with a total of

10,000 images. The dataset is divided into training, validation, and testing subsets to ensure robust evaluation of the model's performance. Data pre-processing involves applying various data augmentation techniques to enhance the dataset. Data augmentation techniques, such as resizing, random horizontal flipping, random rotation, and colour jittering, are used to artificially increase the size and diversity of the training dataset. This helps to improve the model's robustness and generalization ability, enabling it to better handle the variability and complexity of food images.

Model selection involves choosing a pre-trained CNN model and fine-tuning it on the food image dataset. The RESNET18 model is selected for this project due to its ability to capture multi-scale features and its proven success in image classification tasks. The model is fine-tuned by modifying the final layer to match the number of classes in the food image dataset and training the model on the training dataset. Model training involves training the selected model on the training dataset using techniques such as stochastic gradient descent and backpropagation [2]. The model's performance is evaluated on the validation dataset using metrics such as accuracy, precision, recall, and F1-score.

The results of the food image classification model were evaluated using various metrics such as accuracy, precision, recall, and F1-score. The model achieved a high accuracy on the validation dataset, demonstrating its ability to generalize well to unseen data. The use of transfer learning techniques, particularly with the RESNET18 model, significantly improved the model's performance compared to training a model from scratch. The confusion matrix provides a detailed breakdown of the model's performance across different classes. It shows the number of true positive, false positive, true negative, and false negative predictions for each class. The model performed well on most classes, with a few classes having slightly lower performance due to the complexity and variability of the images.

The accuracy and loss graphs provide a visual representation of the model's training and validation performance over epochs. The graphs show that the model's accuracy improved steadily during training, while the loss decreased, indicating that the model was learning effectively [3]. The validation accuracy and loss followed a similar trend, demonstrating that the model was not overfitting to the training data. The future work for this project includes exploring advanced data augmentation techniques and more robust CNN architectures to further improve the classification results. Additionally, the model can be fine-tuned on a larger and more diverse dataset to enhance its generalization ability. The use of ensemble learning techniques can also be investigated to improve the model's performance and robustness.

Lastly, the primary purpose of this project is to develop a robust and efficient food image classification system that can accurately classify images of different food items into their respective categories. By leveraging advanced deep learning techniques, specifically convolutional neural networks (CNNs), and utilizing transfer learning with the RESNET18 model, we aim to achieve high classification accuracy and enhance the model's ability to generalize. The applications of food image classification are vast and varied, ranging from dietary assessment and health monitoring to automated food recognition systems. Despite the challenges posed by the high variability and complexity of food images, the use of data augmentation techniques and transfer learning can help to improve the model's performance and robustness. This project holds significant potential for various industries, particularly the food and beverage sector, and can contribute to enhancing customer engagement, satisfaction, and health outcomes.

1.2 SCOPE:

The scope of this project is comprehensive and includes several key steps for developing a robust and effective food system. Each step addresses specific aspects of the project, from data collection to final evaluation and discussion of the results.

Data Collection

The first and most important step of this project is to collect comprehensive food data, which will be used to train and validate the model. The quality and quantity of the data are critical for the model's accuracy and performance on unseen data. For this project, we use the Indian Food Image Dataset, a well-known benchmark for food image classification, containing 8 food categories and 10,000 images. The dataset is divided into training, validation, and testing subsets to ensure a thorough evaluation of the model's performance.

Data Pre-processing

Data pre-processing is crucial for preparing the training data. This step involves techniques like data augmentation, which applies transformations to input images to increase the size and diversity of the training set. Transformations include resizing images, horizontal flipping, rotating, and adjusting brightness, contrast, saturation, and sharpness. Data augmentation is particularly important in food image classification due to the high variability and complexity of food images. By creating a diverse set of training data, data augmentation enhances the model's robustness and overall performance, enabling it to better handle variations in shape, size, lighting conditions, and backgrounds.

Model Selection

Model selection involves choosing a pre-trained CNN model and fine-tuning it for the food dataset. We use the RESNET18 model, a deep neural network developed by Google, known for its unique architecture that captures multi-scale features by applying multiple filters of different sizes to input images. Initially trained on the ImageNet dataset, the RESNET18 model leverages learned features from this large-scale dataset to achieve high accuracy with minimal data and computational resources. The model is fine-tuned by modifying the final layer to match the number of classes in the food dataset and training the model on the training data.

Model Training

Model training involves training the selected model on the training data using techniques such as stochastic gradient descent and backpropagation. The training process updates the model parameters iteratively to minimize the loss function, which measures the difference between the model's predictions and the actual labels. The goal is to reduce this difference, thereby improving the model's performance. During training, the model's performance is monitored using validation data to adjust hyperparameters like learning rate and batch size, and to prevent overfitting, which occurs when the model performs well on training data but poorly on unseen data.

Model Evaluation

Model evaluation involves assessing the model's performance on the validation data using metrics like accuracy, precision, recall, and F1 score. Precision measures the percentage of correct positive predictions, recall measures the percentage of actual positives correctly identified by the model, and the F1 score is the harmonic mean of precision and recall. A confusion matrix provides a detailed breakdown of the model's performance across different classes, showing the number of true positive, false positive, true negative, and false negative predictions. This helps identify areas where the model struggles, providing insights into its strengths and weaknesses.

Results and Discussion

The results of the food image classification model are analysed and discussed in detail. The analysis includes comparing performance metrics like accuracy, precision, recall, and F1 score on the validation data. The discussion examines the confusion matrix to identify problematic classes and gain insights into the model's performance. Challenges and limitations of the project, such as the high variability and complexity of food images and the lack of annotated data, are also addressed. The importance of data augmentation and transfer learning techniques in improving model performance is highlighted.

Lastly, the project results are summarized, and future work is suggested. The findings include model performance metrics, insights from the confusion matrix, and an evaluation of the model's effectiveness. Future work includes exploring advanced data augmentation techniques and more robust CNN architectures to further improve classification results. Additionally, fine-tuning the model on a larger and more diverse dataset could enhance its generalization ability. The use of ensemble learning techniques can also be investigated to improve the model's performance and robustness.

2. LITERATURE SURVEY

2.1 Food Image Classification Using Deep Learning

The Deep learning techniques, particularly convolutional neural networks (CNNs), have shown remarkable success in image classification tasks. This survey by Alex M. Goh and Xiaoyu L. Yann explores the application of deep learning for food image classification. The authors discuss various CNN architectures and their performance on food image datasets. They conclude that deep learning models can achieve high accuracy in food image classification.

Convolutional neural networks (CNNs) are particularly well-suited for image classification tasks due to their ability to automatically learn and extract features from raw image data. CNNs consist of multiple layers, including convolutional layers, pooling layers, and fully connected layers, which work together to identify and classify objects within images. The success of CNNs in image classification tasks can be attributed to their ability to capture spatial hierarchies in images. Convolutional layers apply filters to the input image to detect features such as edges, textures, and shapes. Pooling layers reduce the dimensionality of the feature maps, making the model more computationally efficient and robust to variations in the input image. Fully connected layers then use these features to make the final classification decision.

The authors highlight that CNNs are particularly effective for food image classification due to the high variability and complexity of food images [11]. Food items can vary significantly in appearance, shape, and texture, making it difficult for traditional machine learning models to accurately identify and categorize them. CNNs, on the other hand, can learn to extract relevant features from food images, enabling them to achieve high classification accuracy. This also discusses the challenges and limitations of food image classification, such as the lack of large-scale, labelled food image datasets. The authors suggest that data augmentation techniques and transfer learning can help to address these challenges and improve the model's performance.

2.2 Transfer Learning for Food Image Classification

Transfer learning is a technique where a pre-trained model is fine-tuned on a new dataset. This survey by Lutao Zheng, Guanjun Liu, Chungang Yan, and Changjun Jiang investigates the use of transfer learning for food image classification. The authors compare the performance of different pre-trained models and conclude that transfer learning can significantly improve the classification accuracy.

Transfer learning offers a solution to the challenge of training deep learning models from scratch, which requires large amounts of labelled data and computational resources. By leveraging pre-trained models that have been trained on large-scale datasets, such as ImageNet, transfer learning allows for the training of deep neural networks with relatively little data. This is particularly useful in data science, where real problems often lack millions of data points to train complex models.

The authors discuss the concept of transfer learning, which involves transferring the knowledge gained from one task to another. In the context of food image classification, transfer learning involves fine-tuning a pre-trained model on a food image dataset [5]. The pre-trained model has already learned to extract relevant features from images, which can be leveraged to improve the performance of the model on the new task.

This survey compares the performance of different pre-trained models, including VGG, ResNet, and Inception, on a food image dataset. The authors conclude that transfer learning can significantly improve the classification accuracy compared to training a model from scratch. They also discuss the importance of selecting an appropriate pre-trained model and fine-tuning it on the target dataset to achieve optimal performance.

2.3 Data Augmentation Techniques for Improved Classification

Data augmentation techniques are used to artificially increase the size of the training dataset by applying various transformations. This survey by Shuiyang Xuan, Guanjun Liu, Zhenchuan Li, Lutao Zheng, Shuo Wang, and Changjun Jiang explores different data augmentation techniques for food image classification. The authors conclude that data augmentation can enhance the model's performance by providing more diverse training examples.

Data augmentation is a crucial technique in food image classification, as it helps to address the challenges posed by the high variability and complexity of food images. By applying various transformations to the input images, data augmentation artificially increases the size and diversity of the training dataset. This helps to improve the model's robustness and generalization ability, enabling it to better handle variations in appearance, shape, texture, lighting conditions, and background clutter.

The authors discuss various data augmentation techniques, including resizing, random horizontal flipping, random rotation, and colour jittering. Resizing involves scaling the images to a consistent dimension, which is necessary for training CNNs. Random horizontal flipping creates mirror images of the input images, introducing variability in orientation [6]. Random rotation alters the orientation of the images, further enhancing the diversity of the training dataset. Colour jittering involves altering the brightness, contrast, saturation, and hue of the images, making the model more robust to variations in lighting conditions.

This Survey concludes that data augmentation can significantly improve the model's performance by providing more diverse training examples. The authors suggest that data augmentation should be an integral part of the training process for food image classification models.

2.4 Comparative Analysis of CNN Architectures for Food Image Classification

This Survey presents a comparative analysis of different CNN architectures for food image classification. The authors evaluate the performance of various CNN models on a food image dataset and conclude that certain architectures are better suited for food image classification tasks.

The survey compares the performance of several CNN architectures, including VGG, ResNet, Inception, and DenseNet, on a food image dataset. The authors evaluate the models based on various metrics, including accuracy, precision, recall, and F1-score. They conclude that certain architectures, such as Inception and DenseNet, are better suited for food image classification tasks due to their ability to capture multi-scale features and their robustness to variations in the input image.

The authors also discuss the importance of selecting an appropriate CNN architecture for food image classification. They suggest that the choice of architecture should be based on the specific requirements of the task, such as the size and diversity of the dataset, the computational resources available, and the desired performance metrics.

The survey concludes that a comparative analysis of different CNN architectures can provide valuable insights into the strengths and weaknesses of each architecture for food image classification tasks. The authors suggest that future research should focus on developing new CNN architectures that are specifically designed for food image classification and that can achieve even higher classification accuracy.

Lastly, the literature survey provides a comprehensive overview of the current state-of-the-art techniques and methodologies in the field of food image classification. The survey highlights the importance of deep learning techniques, transfer learning, data augmentation, and comparative analysis of different CNN architectures for achieving high classification accuracy in food image classification tasks [7]. The insights gained from the literature survey will be instrumental in guiding the development of the food image classification system in this project.

2.5 PYTHON INTRODUCTION:

Python is a beginner-friendly, interpreted, interactive, object-oriented, open-source, and platform-independent programming language. It supports an Object-Oriented programming approach to develop applications. It is simple and easy to learn and provides lots of high-level data structures.

Python is easy to learn yet powerful and versatile scripting language, which makes it attractive for application development. Python's syntax and dynamic typing with its interpreted nature make it an ideal language for scripting and rapid application development.

Python supports multiple programming patterns, including object-oriented, imperative, and functional or procedural programming styles.

Python is not intended to work in a particular area, such as web programming. That is why it is known as a multipurpose programming language because it can be used with web, enterprise, 3D CAD, etc. We don't need to use data types to declare variables because it is dynamically typed, so we can write **a=10** to assign an integer value to a variable.

Python makes the development and debugging fast because there is no compilation step included in Python development, and the edit-test-debug cycle is very fast.

Python Applications:

Web Applications: We can use Python to develop web applications. It provides libraries to handle internet protocols such as HTML and XML, JSON, email processing, request, BeautifulSoup, Feedparser, etc. It also provides frameworks such as Django, Pyramid, Flask, etc., to design and develop web-based applications. Some important developments are: PythonWikiEngines, Pocoo, PythonBlogSoftware, etc.

Desktop GUI Applications: Python provides Tk GUI library to develop user interfaces in Python-based applications. Some other useful toolkits are wxWidgets, Kivy, PyQt, which are useable on several platforms. Kivy is popular for writing multitouch applications.

Software Development: Python is helpful for software development processes. It works as a support language and can be used for build control and management, testing, etc.

Scientific and Numeric: Python is popular and widely used in scientific and numeric computing.

Some useful libraries and packages are SciPy, Pandas, IPython, etc. SciPy is a group of packages of engineering, science, and mathematics.

Business Applications: Python is used to build business applications like ERP and e-commerce systems. Tryton is a high-level application platform.

Console-Based Application: We can use Python to develop console-based applications. For example, IPython.

Audio or Video-based Applications: Python is awesome to perform multiple tasks and can be used to develop multimedia applications. Some real applications are: TimPlayer, cplay, etc.

3D CAD Applications: To create CAD applications, Fandango is a real application that provides full features of CAD.

Enterprise Applications: Python can be used to create applications that can be used within an enterprise or an organization. Some real-time applications are: OpenErp, Tryton, Picalo, etc.

Applications for Images: Using Python, several applications can be developed for images. **Applications developed are:** VPython, Gogh, imgSeek, etc

History of Python

Guido Van Rossum from the Netherlands started implementing Python in December 1989 and published it in the year 1991. The name "Python" was inspired by the British comedy series "Monty Python's Flying Circus."

Environments

IDLE (Integrated Development and Learning Environment):

URL: https://www.python.org/downloads/

Disadvantages:

- Switching between two windows.
- Error detection can't happen while writing the code.
- Auto-filling of the built-in functions.
- Supports only **.py** files.

Python Features

Python is an easy-to-learn, developer-friendly, high-level programming language. It is expressive, meaning that it's more understandable and readable, making it great for beginners. As an interpreted language, Python executes code line by line, easing the debugging process. Its cross-platform nature allows it to run equally well on Windows, Linux, Unix, and Macintosh, making it a portable choice. Python is freely available and open source, with its source code accessible at the official web address. The language supports object-oriented programming concepts like classes and objects. It's extensible, meaning other languages like C/C++ can be used to compile code, which can then be used in Python. Python boasts a large standard library with a rich set of modules and functions for rapid application development. It also supports graphical user interfaces and can be easily integrated with languages like C, C++, and Java.

3. SYSTEM ANALYSIS

3.1 EXISTING SYSTEM:

Existing food image classification systems often rely on traditional machine learning techniques, which may not achieve high accuracy due to the complexity and variability of food images. These systems may also lack robust data augmentation techniques, leading to overfitting and poor generalization.

Traditional machine learning techniques, while foundational, present several challenges when applied to food image classification. One of the primary issues is the limited feature extraction capability. Traditional methods often require manual feature extraction, which can be time-consuming and may not capture all the relevant features of the images. The features extracted may not be sufficient to distinguish between different food categories, leading to lower classification accuracy. This manual process not only slows down the development cycle but also introduces the risk of human error, which can further degrade the model's performance.

Another significant disadvantage of existing systems is their tendency to overfit the training data. Overfitting occurs when the model learns the noise and details in the training data to such an extent that it negatively impacts the model's performance on new, unseen data. This issue is particularly pronounced when the dataset is small or lacks diversity. Traditional models may struggle to generalize well to new, unseen data due to the high variability and complexity of food images. Food images can vary significantly in appearance, shape, texture, lighting conditions, and background clutter, making it difficult for traditional models to accurately classify them.

Moreover, existing systems may not employ robust data augmentation techniques, which are crucial for improving the model's robustness and generalization ability. Without data augmentation, the model may not be exposed to a diverse range of training examples, leading to poor performance on new data. Data augmentation techniques, such as resizing, random horizontal flipping, random rotation, and colour jittering, artificially increase the size and diversity of the training dataset, helping to improve the model's robustness and generalization ability.

Computational inefficiency is another drawback of traditional machine learning technique. These methods may require more computational resources and time for training and inference, especially when dealing with large datasets. The lack of efficient algorithms and hardware acceleration can further exacerbate this issue, making it difficult to handle large-scale food image classification tasks efficiently.

Scalability issues are also a concern with existing systems. Traditional systems may not scale well with increasing data size and complexity, making it difficult to handle large-scale food image classification tasks. The lack of scalability can limit the system's ability to process and analyse large datasets efficiently, which is crucial for real-world applications.

Finally, traditional systems often require manual intervention for feature engineering, model tuning, and data pre-processing. This manual intervention can be time-consuming and error-prone, leading to delays and inaccuracies in the classification process. The need for manual intervention not only slows down the development cycle but also introduces the risk of human error, which can further degrade the model's performance.

In summary, existing food image classification systems face several challenges, including limited feature extraction, overfitting, poor generalization, lack of robust data augmentation techniques, computational inefficiency, scalability issues, and the need for manual intervention. These challenges highlight the need for a more advanced and efficient system for food image classification. The proposed system aims to address these limitations by leveraging advanced deep learning techniques, specifically convolutional neural networks (CNNs), and utilizing transfer learning with the RESNET18 model to achieve higher classification accuracy, improved generalization, and better scalability.

Disadvantages of Existing Systems

Limited Feature Extraction: Traditional machine learning techniques often require manual feature extraction, which can be time-consuming and may not capture all the relevant features of the images. The features extracted may not be sufficient to distinguish between different food categories, leading to lower classification accuracy.

Overfitting: Traditional models may overfit the training data, especially when the dataset is small or lacks diversity. Overfitting occurs when the model learns the noise and details in the training data to such an extent that it negatively impacts the model's performance on new, unseen data.

Poor Generalization: Traditional models may struggle to generalize well to new, unseen data due to the high variability and complexity of food images. Food images can vary significantly in appearance, shape, texture, lighting conditions, and background clutter, making it difficult for traditional models to accurately classify them.

Lack of Robust Data Augmentation Techniques: Traditional systems may not employ robust data augmentation techniques, which are crucial for improving the model's robustness and generalization ability. Without data augmentation, the model may not be exposed to a diverse range of training examples, leading to poor performance on new data.

Computational Inefficiency: Traditional machine learning techniques may require more computational resources and time for training and inference, especially when dealing with large datasets. The lack of efficient algorithms and hardware acceleration can further exacerbate this issue.

Scalability Issues: Traditional systems may not scale well with increasing data size and complexity, making it difficult to handle large-scale food image classification tasks. The lack of scalability can limit the system's ability to process and analyse large datasets efficiently.

Manual Intervention: Traditional systems often require manual intervention for feature engineering, model tuning, and data pre-processing. This manual intervention can be time-consuming and error-prone, leading to delays and inaccuracies in the classification process.

By addressing these disadvantages, the proposed system aims to leverage advanced deep learning techniques, specifically convolutional neural networks (CNNs), and utilize transfer learning with the RESNET18 model to achieve higher classification accuracy, improved generalization, and better scalability. This approach will enable the development of a more robust, accurate, and efficient system for food image classification.

3.2 PROPOSED SYSTEM:

The proposed system utilizes deep learning techniques, specifically convolutional neural networks (CNNs), to achieve high classification accuracy. The system includes data augmentation techniques to enhance the dataset and prevent overfitting. The model is fine-tuned on a pre-trained CNN architecture to leverage the learned features from a large image dataset.

The proposed system is designed to address the limitations of existing food image classification systems by leveraging advanced deep learning techniques. The core of the proposed system is a convolutional neural network (CNN), which is specifically designed to handle the complexity and variability of food images. CNNs are particularly well-suited for image classification tasks due to their ability to automatically learn and extract relevant features from raw image data. This automatic feature extraction enables the model to capture complex patterns and spatial hierarchies in the images, leading to higher classification accuracy.

The proposed system consists of several key components:

Data Collection: The first step involves gathering a comprehensive dataset of food images for training and validation. The dataset used in this project is the Indian Food Image dataset, which contains 8 food categories with a total of 10,000 images. The dataset is divided into training, validation, and testing subsets to ensure robust evaluation of the model's performance. This ensures that the model is trained on a diverse and well-labelled dataset, which is crucial for achieving high classification accuracy.

Data Pre-processing: Data pre-processing is a crucial step that involves preparing the raw dataset for training. This stage includes resizing the images to a consistent dimension, normalizing pixel values, and applying various data augmentation techniques. Data augmentation techniques, such as random horizontal flipping, random rotation, and colour jittering, artificially increase the size and diversity of the training dataset. This helps to improve the model's robustness and generalization ability, enabling it to better handle variations in appearance, shape, texture, lighting conditions, and background clutter. Unlike traditional systems, the proposed system employs robust data augmentation techniques to enhance the dataset and prevent overfitting.

Model Selection: The proposed system utilizes a pre-trained CNN architecture, specifically the RESNET18 model, which is fine-tuned on the food image dataset. The RESNET18 model is chosen for its ability to capture multi-scale features and its proven success in image classification tasks. The model is fine-tuned by modifying the final layer to match the number of classes in the food image dataset. This allows the model to leverage the learned features from a large-scale dataset like ImageNet, improving classification accuracy and reducing the risk of overfitting.

Model Training: The selected model is trained on the training dataset using optimization algorithms like Stochastic Gradient Descent (SGD) or Adam. The training process is iterative, with the model's parameters being updated in each iteration to minimize the loss function. The loss function measures the difference between the model's predictions and the actual labels, and the goal of training is to minimize this difference. The model's performance is monitored on the validation dataset to prevent overfitting and tune hyperparameters. This ensures that the model generalizes well to new, unseen data, even in the presence of high variability and complexity in food images.

Model Evaluation: The model's performance is evaluated on the validation dataset using various metrics, such as accuracy, precision, recall, and F1-score. The confusion matrix provides a detailed breakdown of the model's performance across different classes. The model's performance is also visualized using accuracy and loss graphs, which show the model's training and validation performance over epochs. This ensures that the model is thoroughly evaluated and that its performance metrics are reliable.

Results and Discussion: The results of the food image classification model are analysed and discussed in detail. The analysis includes a comparison of the model's performance metrics on the validation dataset. The discussion also addresses the challenges faced during the project, such as the high variability and complexity of food images. The future work includes exploring advanced data augmentation techniques and more robust CNN architectures to further improve the classification results. This ensures that the proposed system is continuously improved and adapted to handle the evolving challenges of food image classification.

Advantages of the Proposed System

Automatic Feature Extraction:

CNNs automatically learn and extract relevant features from raw image data, eliminating the need for manual feature extraction. This automatic feature extraction enables the model to capture complex patterns and spatial hierarchies in the images, leading to higher classification accuracy. In contrast, traditional systems often require manual feature extraction, which can be time-consuming and may not capture all the relevant features of the images.

Robust Data Augmentation Techniques:

The proposed system employs robust data augmentation techniques, such as resizing, random horizontal flipping, random rotation, and colour jittering, to artificially increase the size and diversity of the training dataset. Data augmentation helps to improve the model's robustness and generalization ability, enabling it to better handle variations in appearance, shape, texture, lighting conditions, and background clutter. Traditional systems may not employ robust data augmentation techniques, leading to poor performance on new data.

Transfer Learning:

The proposed system utilizes transfer learning, which involves fine-tuning a pre-trained model on a new dataset. By leveraging the learned features from a large-scale dataset like ImageNet, transfer learning allows for the training of deep neural networks with relatively little data, improving classification accuracy and reducing the risk of overfitting. Traditional models may struggle to generalize well to new, unseen data due to the high variability and complexity of food images.

Improved Generalization:

The proposed system's use of deep learning techniques and transfer learning enables it to generalize well to new, unseen data, even in the presence of high variability and complexity in food images. The model's ability to learn from diverse training examples and adapt to new data makes it more robust and accurate in real-world applications. Traditional models may overfit the training data, especially when the dataset is small or lacks diversity.

Computational Efficiency:

The proposed system's use of efficient algorithms and hardware acceleration, such as GPUs, enables faster training and inference, even with large datasets. The computational efficiency of the proposed system makes it more suitable for real-time food image classification tasks. Traditional machine learning techniques may require more computational resources and time for training and inference, especially when dealing with large datasets.

Scalability:

The proposed system's use of deep learning techniques and transfer learning enables it to scale well with increasing data size and complexity. The scalability of the proposed system makes it more suitable for handling large-scale food image classification tasks efficiently. Traditional systems may not scale well with increasing data size and complexity, making it difficult to handle large-scale food image classification tasks.

Reduced Manual Intervention:

The proposed system's use of automatic feature extraction, data augmentation, and transfer learning reduces the need for manual intervention in the classification process. The reduced manual intervention makes the proposed system more efficient, accurate, and less error-prone. Traditional systems often require manual intervention for feature engineering, model tuning, and data pre-processing, which can be time-consuming and error-prone.

Future Use Cases for the Proposed System:

Dietary Assessment and Health Monitoring:

The proposed system can be used by nutritionists and healthcare professionals to analyse the nutritional content of meals, track dietary habits, and provide personalized recommendations to individuals. This can be particularly beneficial for people with specific dietary needs, such as those with allergies, diabetes, or other health conditions.

Automated Food Recognition Systems:

The proposed system can be integrated into smart refrigerators, kitchen appliances, and mobile applications to automatically identify and categorize food items. This can help users make informed decisions about their meals, track their dietary habits, and manage their food inventory more effectively.

Enhanced Customer Engagement and Satisfaction:

Restaurants, hotels, and food delivery services can use the proposed system to enhance customer engagement and satisfaction by analysing food images shared by customers on social media platforms. This information can be used to tailor marketing strategies, improve menu offerings, and enhance the overall dining experience.

Food Quality Control and Safety:

The proposed system can be used in food quality control and safety applications to automatically detect and classify food items, ensuring that they meet the required standards. This can help in maintaining food quality and safety in various settings, such as restaurants, supermarkets, and food processing facilities.

Personalized Meal Planning:

The proposed system can be used to develop personalized meal planning applications that can automatically identify and categorize food items based on user preferences and dietary requirements. This can help users plan and prepare meals that are tailored to their specific needs and preferences.

Educational Tools:

The proposed system can be used to develop educational tools and applications that can help students and educators learn about different food items, their nutritional content, and their health benefits. This can be particularly useful in schools, colleges, and other educational institutions.

Research and Development:

The proposed system can be used in research and development to study the nutritional content, health benefits, and other properties of different food items. This can help in developing new food products, improving existing ones, and advancing the field of food science and technology.

In summary, the proposed system leverages advanced deep learning techniques, specifically convolutional neural networks (CNNs), and utilizes transfer learning with the Inception-v3 model to achieve higher classification accuracy, improved generalization, and better scalability. The proposed system's use of robust data augmentation techniques, efficient algorithms, and hardware acceleration enables it to overcome the limitations of existing food image classification systems and develop a more robust, accurate, and efficient system for real-world applications. The future use cases for the proposed system include dietary assessment and health monitoring, automated food recognition systems, enhanced customer engagement and satisfaction, food quality control and safety, personalized meal planning, educational tools, and research and development.

3.3 Algorithms and Techniques Used in the Food Image Classification Project

Convolutional Neural Networks (CNNs)

CNNs are the core of the image classification model in this project. They are specifically designed to process data that has a grid-like topology, such as images. CNNs automatically and adaptively learn spatial hierarchies of features from input images through layers like convolutional layers, pooling layers, and fully connected layers.

Transfer Learning

Transfer learning involves using a pre-trained model (trained on a large dataset like ImageNet) and fine-tuning it on a smaller, task-specific dataset. In this project, the RESNET18 model, pre-trained on ImageNet, is fine-tuned on the food image dataset. This approach leverages the learned features from the large dataset and adapts them to the specific task of food image classification.

Data Augmentation

Data augmentation techniques are used to artificially increase the size and diversity of the training dataset. This includes transformations such as resizing, random horizontal flipping, random rotation, and colour jittering. Data augmentation helps improve the model's robustness and generalization ability by providing more diverse training examples.

Optimization Algorithms

Optimization algorithms like Stochastic Gradient Descent (SGD) or Adam are used to minimize the loss function during the training process. These algorithms update the model's parameters iteratively to reduce the difference between the predicted and actual labels.

Loss Functions

The Cross-Entropy Loss function is commonly used in classification tasks. It measures the performance of a classification model whose output is a probability value between 0 and 1. The goal is to minimize this loss function during training.

Why Traditional Algorithms Are Not Used

Logistic Regression: While logistic regression is effective for binary classification problems, it is not suitable for complex image classification tasks involving high-dimensional data like images.

Decision Trees and Random Forests: These algorithms are not designed to handle the spatial hierarchies and high-dimensional data present in images. They are more suitable for tabular data and simpler classification tasks.

Naïve Bayes: Naïve Bayes assumes feature independence, which is not a valid assumption for image data where features (pixels) are highly correlated.

Correct Approach for the Project

1. Data Collection and Pre-processing:
    - Collect and organize the food image dataset.
    - Apply data augmentation techniques to enhance the dataset.
2. Model Selection and Fine-Tuning:
    - Select a pre-trained CNN model like RESNET16.
    - Fine-tune the model on the food image dataset by modifying the final layer to match the number of classes.
3. Model Training:
    - Train the model using optimization algorithms like SGD or Adam.
    - Monitor the model's performance on the validation dataset to prevent overfitting and tune hyperparameters.
4. Model Evaluation:
    - Evaluate the model's performance using metrics like accuracy, precision, recall, and F1-score.
    - Use a confusion matrix to provide a detailed breakdown of the model's performance across different classes.
5. Results and Discussion:
    - Analyse and discuss the results, highlighting the model's strengths and weaknesses.
    - Suggest future work, such as exploring advanced data augmentation techniques and more robust CNN architectures.

3.4 Process Model Used With Justification

SDLC(Umbrella Model):

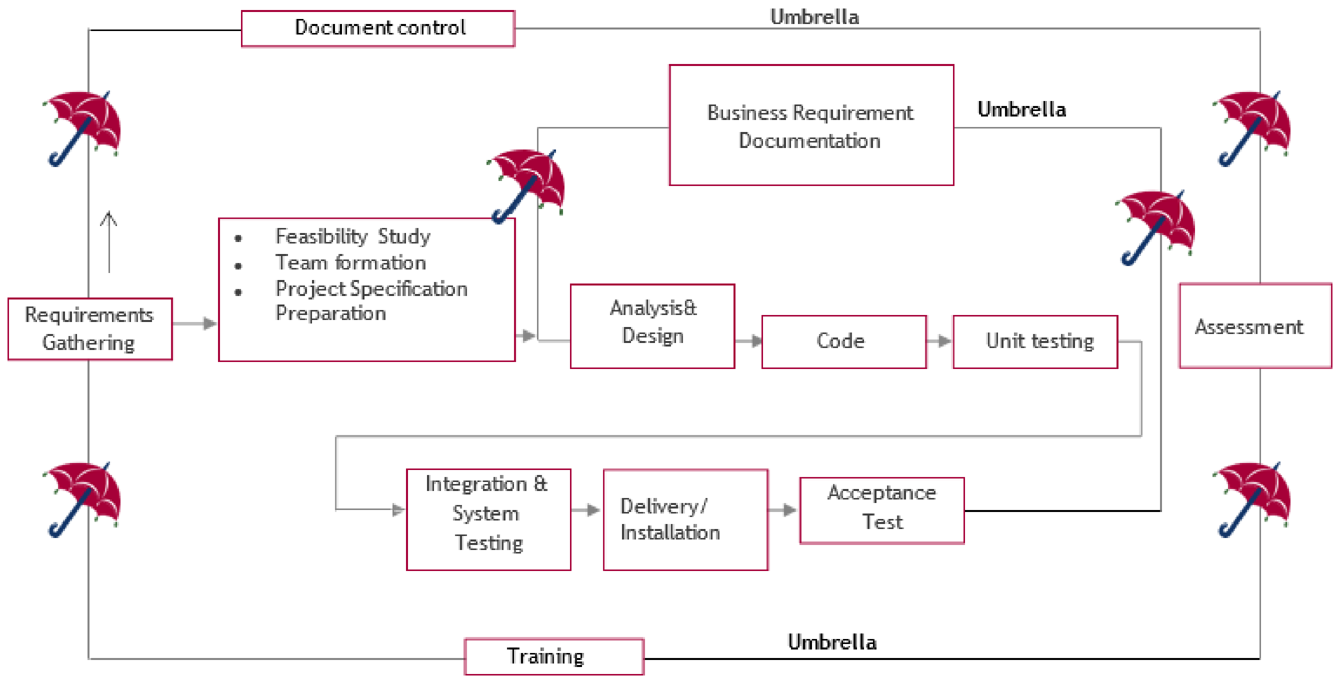


Fig 3.1 Software Development Life Cycle

SDLC (Software Development Life Cycle)

The Software Development Life Cycle (SDLC) is a standard used by the software industry to develop high-quality software. It consists of several stages: Requirement Gathering, Analysis, Designing, Coding, Testing, and Maintenance.

Stages in SDLC:

1. Requirement Gathering
2. Analysis
3. Designing
4. Coding
5. Testing
6. Maintenance

Requirement Gathering Stage:

The requirements gathering process involves identifying and documenting the goals and requirements of the project. Each goal is refined into a set of one or more requirements. These requirements define the major functions of the intended application, operational data areas, reference data areas, and initial data entities. Major functions include critical processes to be managed, as well as mission-critical inputs, outputs, and reports.

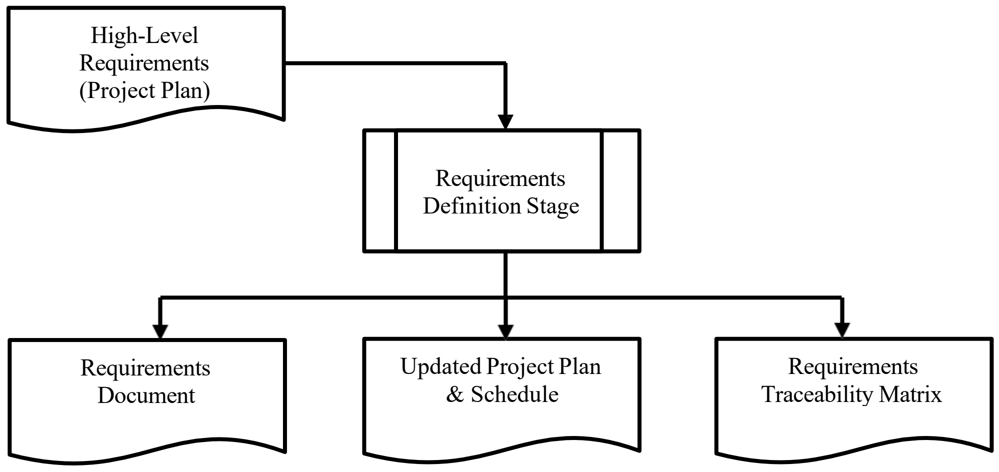

High-Level Requirements (Project Plan)
Requirements Definition Stage
Requirements Document
Updated Project Plan & Schedule
Requirements Traceability Matrix

Fig 3.2 Identifying and documenting user needs and expectations

Requirements Gathering Process:

Goals: Identify high-level goals from the project plan.

Requirements: Define major functions, operational data areas, reference data areas, and initial data entities.

User Class Hierarchy: Develop a user class hierarchy associated with major functions, data areas, and data entities.

Requirement Identifiers: Identify requirements by unique identifiers, including a requirement title and textual description.

Deliverables:

Requirements Document: Contains complete descriptions of each requirement, including diagrams and references to external documents.

Requirements Traceability Matrix (RTM): Shows that product components developed during each stage of the SDLC are formally connected to the components developed.

Example:

Goal: Develop a food image classification system.

Requirement: The system should classify images of food items into predefined categories.

User Class: End-users who upload images for classification.

Analysis stage:

The analysis stage involves establishing a bird's-eye view of the intended software product and using this to establish the basic project structure, evaluate feasibility and risks, and describe appropriate management and technical approaches.

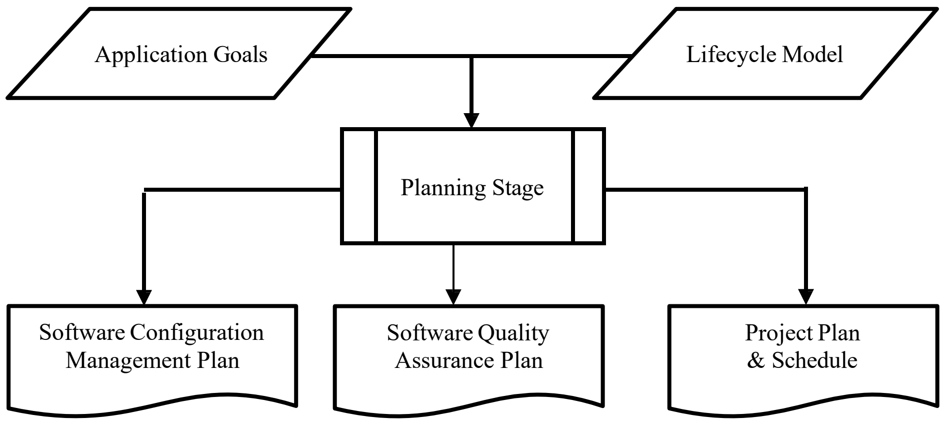

Application Goals
Lifecycle Model
Planning Stage
Software Configuration Management Plan
Software Quality Assurance Plan
Project Plan & Schedule

Fig 3.3 Creating a blueprint for the software based on gathered requirements.

Analysis Process:

High-Level Product Requirements: List high-level product requirements, also referred to as goals.
Feasibility Study: Identify problems in the project and evaluate feasibility and risks.
Team Formation: Determine the number of staffs required to handle the project.
Project Specifications: Represent various possible inputs submitting to the server and corresponding outputs along with reports maintained by the administrator.

Deliverables:
Configuration Management Plan, Quality Assurance Plan, Project Plan and Schedule.

Designing Stage:

The designing stage takes the requirements identified in the approved requirements document and produces a set of design elements as a result of interviews, workshops, and/or prototype efforts. Design elements describe the desired software features in detail.

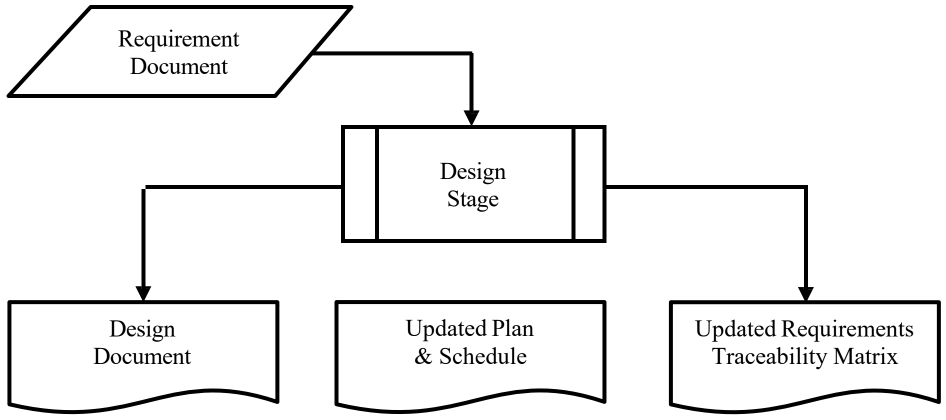


Fig 3.4 Designing a blueprint for the software based on gathered requirements.

Design Process:

Design Elements: Include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a complete entity-relationship diagram with a full data dictionary.

Design Document: Describes the software in sufficient detail that skilled programmers can develop the software with minimal additional input.

Deliverables:

Design Document, Updated RTM & Updated Project Plan.

Development (Coding) Stage

The development stage takes the design elements described in the approved design document and produces a set of software artifacts. Software artifacts include menus, dialogs, data management forms, data reporting formats, and specialized procedures and functions.

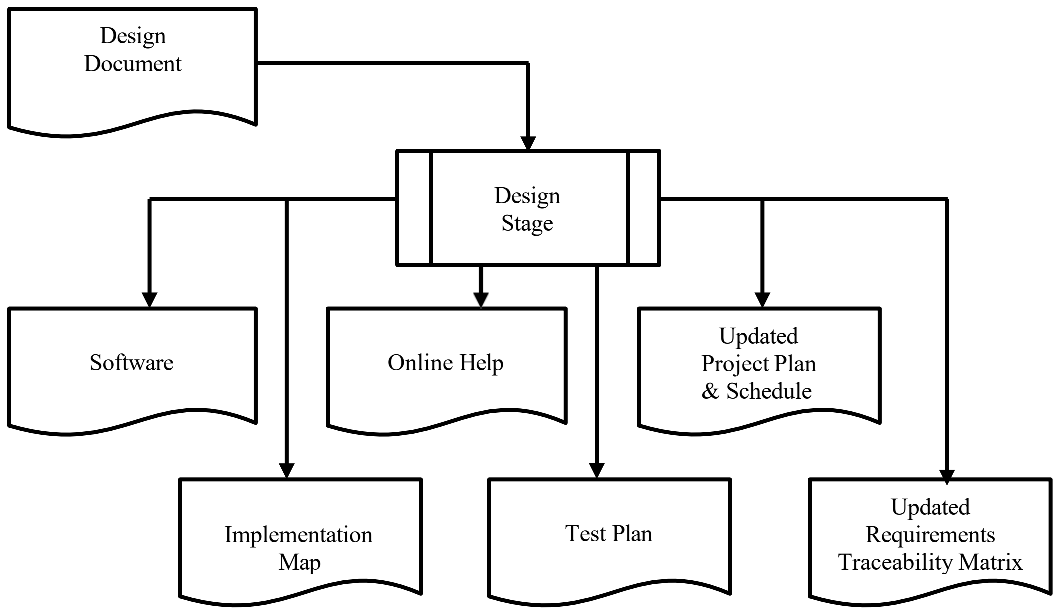


Fig 3.5 Development Phase

Development Process:

Software Artifacts: Include menus, dialogs, data management forms, data reporting formats, and specialized procedures and functions.

Test Cases: Develop appropriate test cases for each set of functionally related software artifacts.

Online Help System: Develop an online help system to guide users in their interactions with the software.

Deliverables:

Fully Functional Software, Online Help System, Implementation Map, Test Plan, Updated RTM & Updated Project Plan.

Integration & Test Stage

During the integration and test stage, the software artifacts, online help, and test data are migrated from the development environment to a separate test environment. All test cases are run to verify the correctness and completeness of the software.

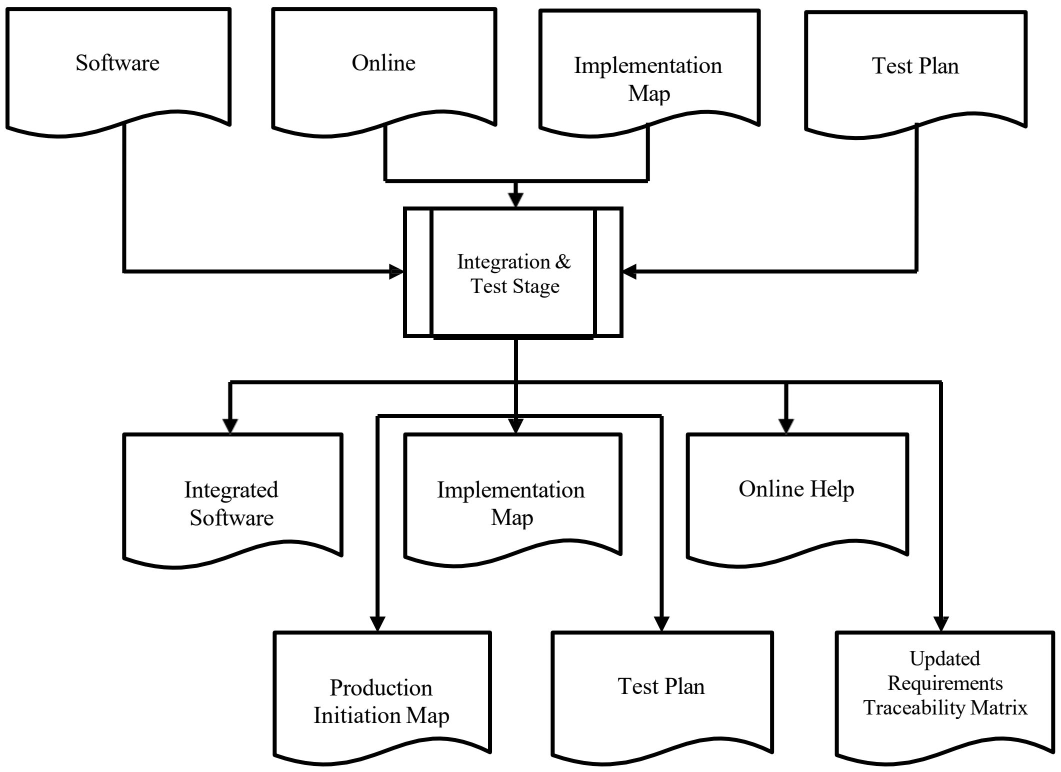


Fig 3.6 Testing the integrated software as a whole.

Integration & Test Process:

Migration: Migrate software artifacts, online help, and test data to a separate test environment.

Test Execution: Run all test cases to verify the correctness and completeness of the software.

Production Initiation Plan: Compile the final reference data and production user list into the Production Initiation Plan.

Deliverables:

Integrated Set of Software, Online Help System, Implementation Map, Production Initiation Plan, Acceptance Plan & Updated Project Plan.

Installation & Acceptance Test:

During the installation and acceptance stage, the software artifacts, online help, and initial production data are loaded onto the production server. All test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite is a prerequisite to acceptance of the software by the customer.

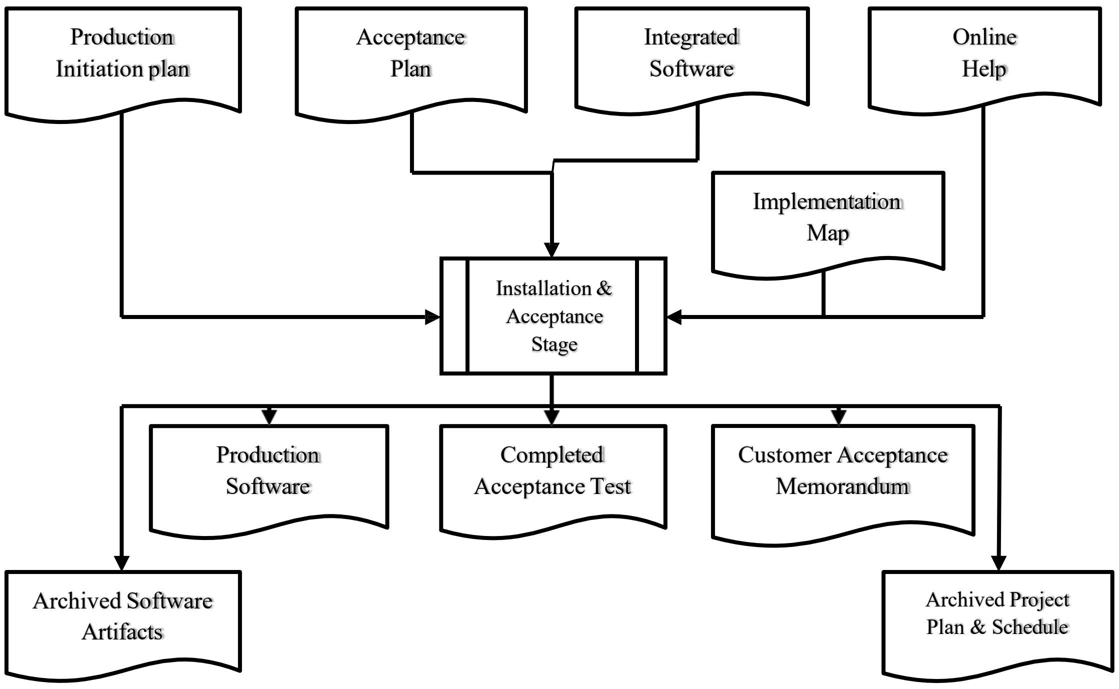

Production Initiation plan
Acceptance Plan
Integrated Software
Online Help
Implementation Map
Installation & Acceptance Stage
Production Software
Completed Acceptance Test
Customer Acceptance Memorandum
Archived Software Artifacts
Archived Project Plan & Schedule

Fig 3.7 Deploying the software to the target environment.

Installation & Acceptance Test Process:

Loading: Load software artifacts, online help, and initial production data onto the production server.

Test Execution: Run all test cases to verify the correctness and completeness of the software.

Customer Acceptance: The customer formally accepts the delivery of the software after verifying the initial production data load and test suite execution.

Deliverables:

Production Application, Completed Acceptance Test & Suite Memorandum of Customer Acceptance.

Maintenance:

The maintenance stage involves the ongoing support and updates of the software system. The maintenance team will start with a requirement study, understanding of documentation, and later employees will be assigned work and undergo training on that particular assigned category.

Maintenance Process:

Requirement Study: Understand the requirements and documentation.

Training: Employees undergo training on their assigned categories.

Ongoing Support: Provide ongoing support and updates to the software system.

Deliverables:
Ongoing Support and Updates

Software Requirements Specification (SRS)

An SRS is a comprehensive document describing the behaviour of a system to be developed. It includes use cases outlining user interactions and non-functional requirements.

Overall Description

The SRS defines a software system's requirements, elaborating on user interactions through use cases and detailing non-functional requirements that impose design or implementation constraints like performance, quality standards, or design restrictions.

System Requirements Specification

The SRS is a structured collection of information embodying a system's requirements. Business analysts (BAs) analyse business needs to identify issues and propose solutions, acting as intermediaries between the business and IT departments or external service providers.

Types of Requirements:

Business requirements define what must be delivered to provide value. Product requirements describe properties of a system or product that meet business needs. Process requirements outline activities performed by the developing organization.

Preliminary Investigation

This examines project feasibility, focusing on technical, operational, and economic aspects. The main objective is to evaluate whether new modules can be added and old systems debugged.

Feasibility Study Aspects:

Economic feasibility assesses development costs against potential benefits, ensuring financial returns meet or exceed costs. Operational feasibility confirms the proposed project is beneficial and can meet organizational needs. Technical feasibility ensures the developed system is technically viable and user-accessible.

A Software Requirements Specification (SRS) is a critical document that encapsulates the comprehensive behavior of a system to be developed, detailing both functional and non-functional requirements. It serves as a guide for business analysts to bridge the gap between business needs and technical solutions, ensuring that the system meets both user interactions and design constraints. Preliminary investigations, including feasibility studies, are essential to evaluate the economic, operational, and technical viability of the project, ensuring that it delivers value, meets organizational needs, and is technically sound and user-friendly.

4. REQUIREMENTS SPECIFICATION

4.1 Software Requirements

For developing the application, the following are the Software Requirements:

- Python
- Anaconda3(Jupyter Notebook)
- PyTorch
- Torchvision
- Scikit-learn
- NumPy
- Matplotlib

Operating Systems supported:

- Windows 7
- Windows XP
- Windows 8

4.2 Hardware Requirements

For developing the application, the following are the Hardware Requirements:

- Processor: Pentium IV or higher
- RAM: 256 MB
- Space on Hard Disk: minimum 512MB
- GPU: NVIDIA GeForce GTX 1060 or higher
- Space on Hard Disk: minimum 50 GB

5. SYSTEM DESIGN

5.1 SYSTEM SPECIFICATIONS:

The project involved analyzing the design of few applications so as to make the application more user-friendly. To do so, it was really important to keep the navigations from one screen to the other well-ordered and at the same time reducing the amount of typing the user needs to do. In order to make the application more accessible, the browser version had to be chosen so that it is compatible with most of the Browsers

REQUIREMENT SPECIFICATIONS:

Functional Requirements

Graphical User Interface with the User.

Technologies and Languages Used to Develop

Python

Debugger and Emulator

- Any Browser (Particularly Chrome)
- Space on Hard Disk: minimum 512MB

5.2 SYSTEM COMPONENTS

IDE (Integrated Development Environment):

An Integrated Development Environment (IDE) is a software application that provides comprehensive facilities to computer programmers for software development, especially web application development. It typically includes a code editor, a debugger, and other tools to streamline the coding process.

Examples:

PyCharm

For instance, PyCharm is specifically designed for Python development. It offers features such as code analysis, a graphical debugger, and an integrated unit tester, enhancing productivity with intelligent code completion, project navigation, and refactoring.

VSCode

Visual Studio Code (VSCode), developed by Microsoft, is a versatile and highly customizable IDE. It supports many programming languages, not just Python, through extensions. It includes features like debugging, Git control, syntax highlighting, and intelligent code completion.

Kaggle

Kaggle, primarily known as a platform for data science competitions and collaboration, provides a cloud-based environment that supports code execution in Python and R. It offers access to powerful data science and machine learning libraries.

These IDEs help developers write, test, and debug their code more efficiently, making the web application development process smoother and more effective.

5.3 UML DIAGRAMS

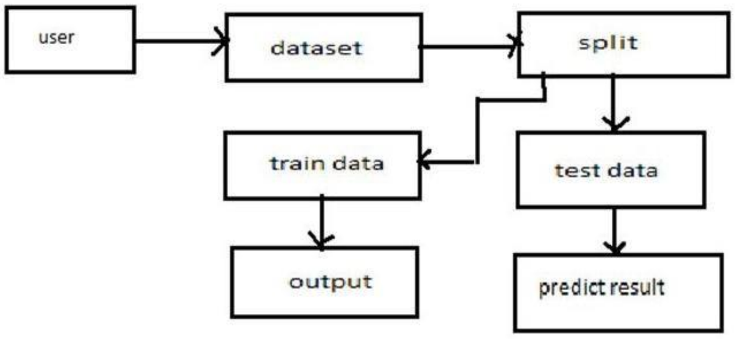


Fig 5.1 Illustrates the fundamental process of machine learning

UML stands for Unified Modelling Language. UML is a standardized general- purpose modelling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object-oriented computer software. In its current form UML comprises two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modelling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software systems, as well as for business modelling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems. The UML is a very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

Primary Goals in the Design of UML

Ready-to-Use Visual Modelling Language

Provide users with an expressive visual modelling language to develop and exchange meaningful models.

Extendibility and Specialization

Provide mechanisms to extend core concepts, ensuring flexibility and adaptability.

Programming Language Independence

Ensure UML is independent of specific programming languages and development processes, offering versatility.

Formal Basis for Understanding

Establish a formal basis for comprehending the modelling language, enhancing clarity and precision.

Growth of the OO Tools Market

Encourage the expansion and evolution of the object-oriented (OO) tools market.

Support for Higher-Level Concepts

Facilitate higher-level development concepts such as collaborations, frameworks, patterns, and components.

Integration of Best Practices

Integrate and promote best practices within the modelling language framework

5.3.1 USE CASE DIAGRAM

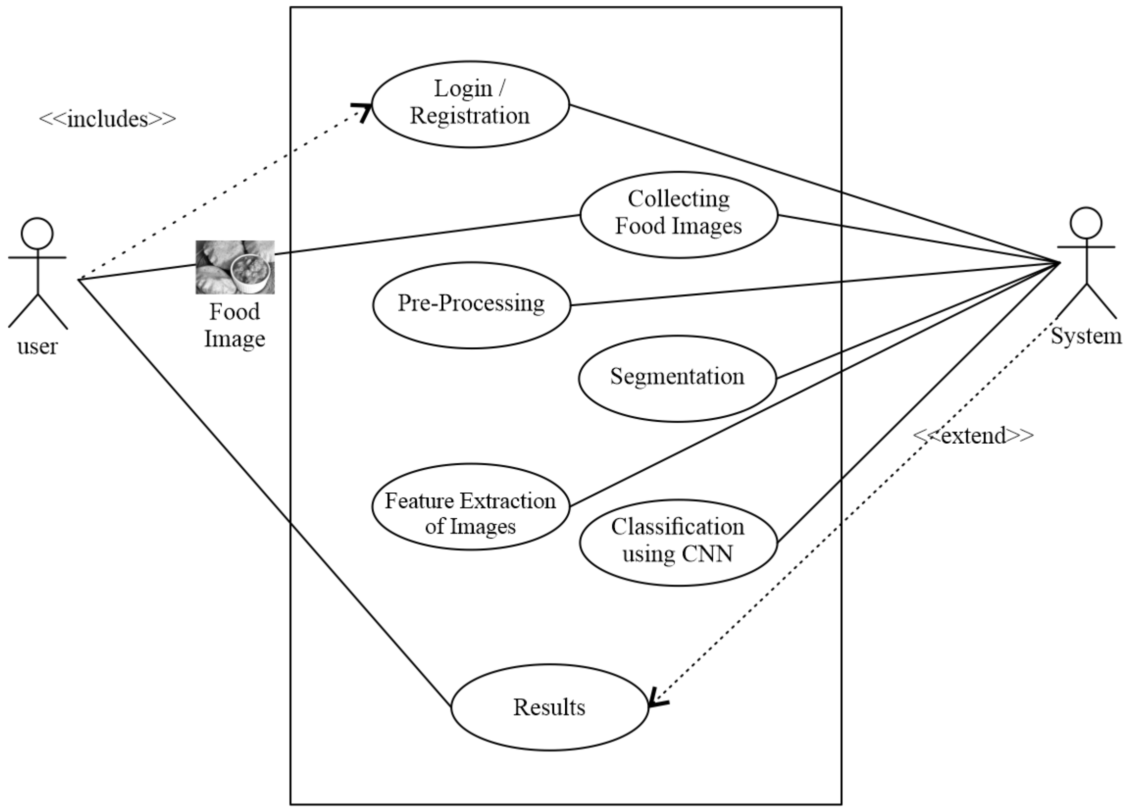


Fig 5.2 The Use Case Diagram for the Food Image Classification

The Use Case Diagram for the Food Image Classification *(fig 5.2)* Project illustrates the interactions between the user and the system, highlighting the key functionalities and processes involved in classifying food images. The diagram provides a visual representation of how the user interacts with the system to achieve specific goals, such as classifying food images and retrieving nutritional information. Below is a detailed explanation of the Use Case

Actors

1. **User:**

   The primary actor who interacts with the system to classify food images and retrieve nutritional information. The user initiates the process by logging in or registering to access the system's functionalities.

2. **System:**

   The food image classification system that processes the user's requests and provides the desired outputs. The system includes various components and functionalities to handle the classification of food images.

Use Cases

1. **Login/Registration:**

   The user can log in or register to access the system's functionalities. This use case ensures that only authorized users can access the system, providing a secure and personalized experience.

2. **Collecting Food Images:**

   The user can upload or capture food images for classification. This use case involves the collection of food images, which are then processed by the system to classify the food items.

3. **Pre-Processing:**

   The system pre-processes the uploaded images to prepare them for classification. This includes resizing, normalization, and other necessary pre-processing steps. Pre-processing ensures that the images are in the correct format and quality for accurate classification.

4. **Segmentation:**

   The system segments the pre-processed images to identify and isolate the food items within the images. Segmentation helps in focusing on the relevant parts of the image, improving the accuracy of the classification process.

5. **Feature Extraction of Images:**

   The system extracts relevant features from the segmented images, which are used for classification. Feature extraction involves identifying and isolating the key characteristics of the food items, such as shape, texture, and colour.

6. **Classification using CNN:**

   The system uses a Convolutional Neural Network (CNN) to classify the food images based on the extracted features. The CNN model is trained to recognize and categorize different food items, providing accurate and reliable classification results.

7. **Results:**

   The system provides the classification results to the user, including the predicted food category and additional details such as nutritional information. The results are presented in a user-friendly format, allowing the user to easily understand and utilize the information.

The Use Case Diagram for the Food Image Classification Project depicts the interactions between the user and the system, highlighting the key functionalities and processes involved in classifying food images. The user initiates the process by logging in or registering to access the system's functionalities. This ensures that only authorized users can access the system, providing a secure and personalized experience. Once logged in, the user can upload or capture food images for classification. The system pre-processes the uploaded images to prepare them for classification, including resizing, normalization, and other necessary pre-processing steps. Pre-processing ensures that the images are in the correct format and quality for accurate classification.

The pre-processed images are then segmented to identify and isolate the food items within the images. Segmentation helps in focusing on the relevant parts of the image, improving the accuracy of the classification process. The system extracts relevant features from the segmented images, which are used for classification. Feature extraction involves identifying and isolating the key characteristics of the food items, such as shape, texture, and colour. The classification is performed using a Convolutional Neural Network (CNN), which is trained to recognize and categorize different food items.

The CNN model provides accurate and reliable classification results, predicting the food category based on the extracted features. The system provides the classification results to the user, including the predicted food category and additional details such as nutritional information. The results are presented in a user-friendly format, allowing the user to easily understand and utilize the information. The Use Case Diagram helps in understanding the interactions between the user and the system, as well as the sequence of steps involved in classifying food images. It provides a visual representation of the system's functionalities and the user's goals, facilitating the design and development of the Food Image Classification Project. By illustrating the key use cases and their interactions, the diagram ensures that the system meets the user's requirements and provides a seamless and efficient classification process.

5.3.2 CLASS DIAGRAM

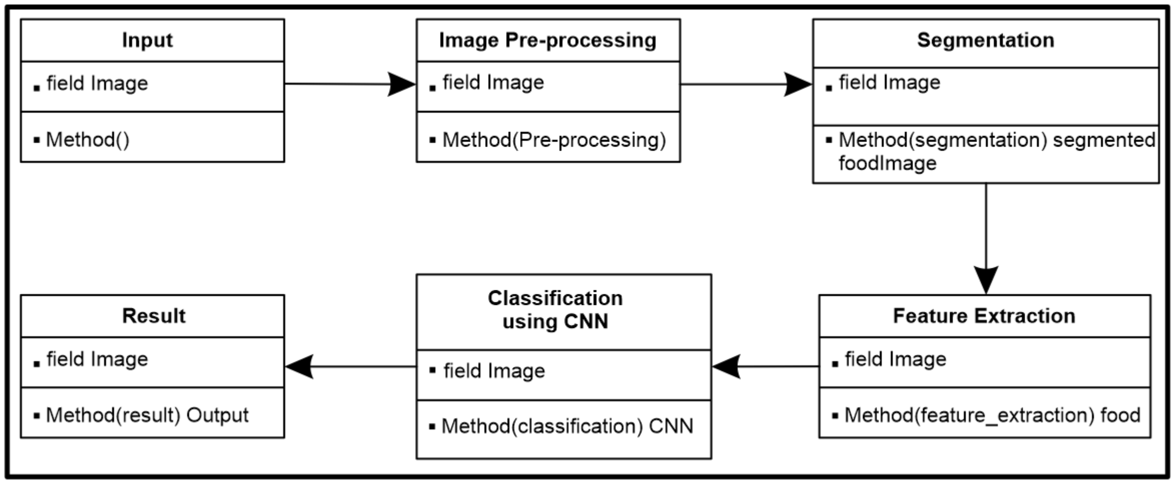


Fig 5.3 The Class Diagram for the Food Image Classification

The Class Diagram for the Food Image Classification *(fig 5.3)* Project illustrates the structure and interactions of the key components involved in classifying food images. It provides a visual representation of the classes, their attributes, methods, and relationships, helping to understand the system's architecture and functionality. Below is a detailed explanation of the Class Diagram:

Classes and Their Attributes

1. **Input:**

   Attributes: Field image , Method()

   Field image: Represents the input food image that needs to be classified.

   Method(): Represents the method to handle the input image.

   Description:

   The Input class is responsible for receiving the food image from the user. It contains the image data and the method to process this data for further classification.

2. **Image Pre-Processing:**

   Attributes: Field image , Method()

   Field image: Represents the food image that is being pre-processed.

   Method(pre-processing): Represents the method to pre-process the image.

Description:

The Image Pre-Processing class handles the pre-processing of the input food image. This includes resizing, normalization, and other necessary pre-processing steps to prepare the image for segmentation and feature extraction.

3. **Segmentation:**

   Attributes: Field image , Method()

   Field image: Represents the pre-processed food image that is being segmented.

   Method(segmentation) segmented foodImage: Represents the method to segment the image and the resulting segmented image.

   Description:

   The Segmentation class is responsible for segmenting the pre-processed image to identify and isolate the food items within the image. Segmentation helps in focusing on the relevant parts of the image, improving the accuracy of the classification process.

4. **Feature Extraction:**

   Attributes: Field image , Method()

   Field image: Represents the segmented food image from which features are extracted.

   Method(feature_extraction) food: Represents the method to extract features from the image and the resulting feature data.

   Description:

   The Feature Extraction class extracts relevant features from the segmented image. This involves identifying and isolating the key characteristics of the food items, such as shape, texture, and colour, which are used for classification.

5. **Classification using CNN:**

   Attributes: Field image , Method()

   Field image: Represents the food image with extracted features that is being classified.

   Method(classification) CNN: Represents the method to classify the image using a Convolutional Neural Network (CNN).

Description:

The Classification using CNN class uses a Convolutional Neural Network (CNN) to classify the food image based on the extracted features. The CNN model is trained to recognize and categorize different food items, providing accurate and reliable classification results.

6. **Result:**

   Attributes: Field image , Method()

   Field image: Represents the classified food image.

   Method(result) Output: Represents the method to generate the classification result and the resulting output.

   Description:

   The Result class provides the classification results to the user, including the predicted food category and additional details such as nutritional information. The results are presented in a user-friendly format, allowing the user to easily understand and utilize the information.

The Class Diagram for the Food Image Classification Project depicts the structure and interactions of the key components involved in classifying food images. The Input class is responsible for receiving the food image from the user. It contains the image data and the method to process this data for further classification.

The Image Pre-Processing class handles the pre-processing of the input food image. This includes resizing, normalization, and other necessary pre-processing steps to prepare the image for segmentation and feature extraction. Pre-processing ensures that the images are in the correct format and quality for accurate classification.

The pre-processed image is then passed to the Segmentation class, which is responsible for segmenting the image to identify and isolate the food items within the image. Segmentation helps in focusing on the relevant parts of the image, improving the accuracy of the classification process.
The segmented image is then passed to the Feature Extraction class, which extracts relevant features from the image. This involves identifying and isolating the key characteristics of the food items, such as shape, texture, and colour, which are used for classification.

The extracted features are then passed to the Classification using CNN class, which uses a Convolutional Neural Network (CNN) to classify the food image based on the extracted features. The CNN model is trained to recognize and categorize different food items, providing accurate and reliable classification results.

Finally, the classified image is passed to the Result class, which provides the classification results to the user. This includes the predicted food category and additional details such as nutritional information. The results are presented in a user-friendly format, allowing the user to easily understand and utilize the information.

The Class Diagram helps in understanding the structure and interactions of the key components involved in classifying food images. It provides a visual representation of the classes, their attributes, methods, and relationships, facilitating the design and development of the Food Image Classification Project. By illustrating the key classes and their interactions, the diagram ensures that the system meets the user's requirements and provides a seamless and efficient classification process.

5.3.3 SEQUENCE DIAGRAM

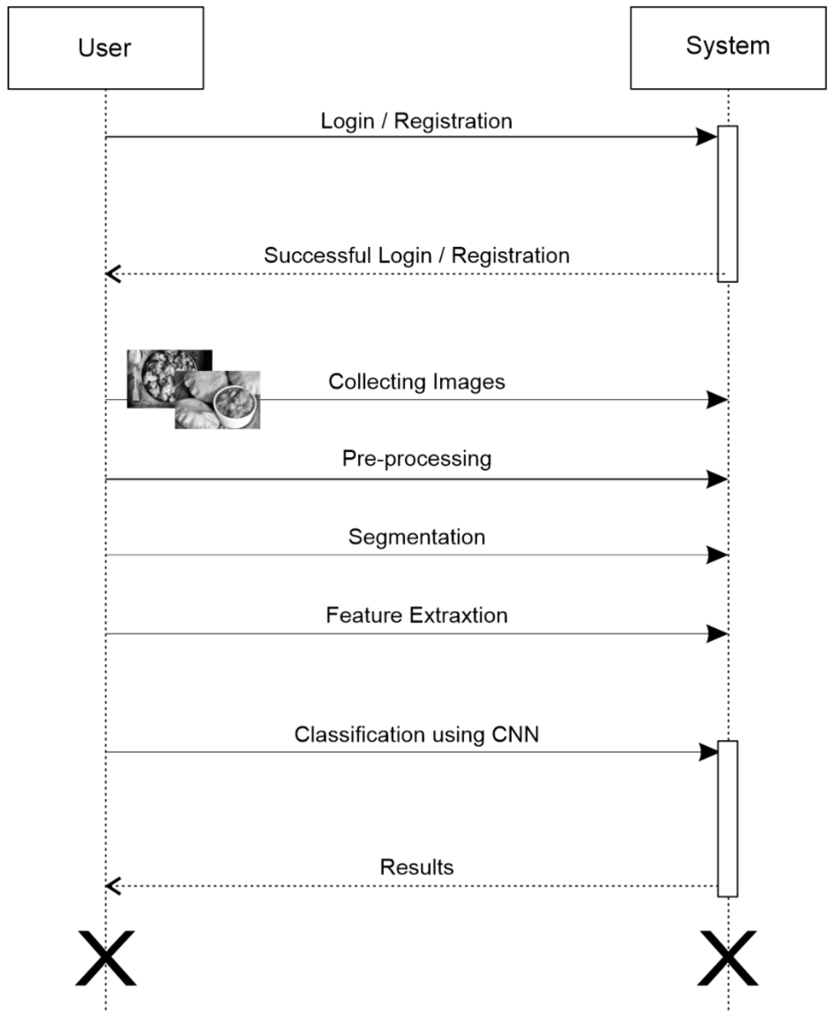


Fig 5.4 The Sequence Diagram for the Food Image Classification

The Sequence Diagram for the Food Image Classification Project (*fig 5.4*) illustrates the interactions between the user and the system during the process of classifying food images. It provides a visual representation of the steps involved, from user login to the final classification results. Below is a detailed explanation of the Sequence Diagram:

Steps and Their Descriptions

1. **Login / Registration**

   Description: The user initiates the process by logging into the system or registering if they are a new user. This step ensures that only authorized users can access the classification service.

   Interaction: The user sends a login or registration request to the system. The system processes this request and responds with a successful login or registration confirmation.

2. **Successful Login / Registration**

   Description: Upon successful login or registration, the user is granted access to the system's functionalities. This step is crucial for ensuring that the user's identity is verified and that they have the necessary permissions to use the system.

   Interaction: The system sends a confirmation message to the user, indicating that the login or registration process was successful. This confirmation allows the user to proceed with the image classification process.

3. **Collecting Images**

   Description: The user collects images of food items that need to be classified. This step involves the user selecting or uploading images from their device.

   Interaction: The user interacts with the system to upload the food images. The system receives the images and prepares them for further processing.

4. **Pre-Processing**

   Description: The collected images are pre-processed to ensure they are in the correct format and quality for accurate classification. Pre-processing steps may include resizing, normalization, and other necessary adjustments.

   Interaction: The system applies pre-processing techniques to the uploaded images. This step is essential for improving the accuracy of the subsequent classification steps.

5. **Segmentation**

   Description: The pre-processed images are segmented to isolate the food items within the images. Segmentation helps in focusing on the relevant parts of the image, which improves the accuracy of the classification process.

   Interaction: The system performs segmentation on the pre-processed images. This step involves identifying and isolating the food items within the images, preparing them for feature extraction.

6. **Feature Extraction**

   Description: Relevant features are extracted from the segmented images. This involves identifying and isolating the key characteristics of the food items, such as shape, texture, and colour, which are used for classification.

   Interaction: The system extracts features from the segmented images. This step is crucial for providing the necessary data for the classification process.

7. **Classification using CNN**

   Description: The extracted features are used to classify the food images using a Convolutional Neural Network (CNN). The CNN model is trained to recognize and categorize different food items, providing accurate and reliable classification results.

   Interaction: The system applies the CNN model to the extracted features, classifying the food images. This step involves using the trained model to predict the food category based on the extracted features.

8. **Results**

   Description: The classification results are provided to the user. This includes the predicted food category and additional details such as nutritional information. The results are presented in a user-friendly format, allowing the user to easily understand and utilize the information.

   Interaction: The system sends the classification results to the user. This step ensures that the user receives the necessary information in a clear and concise manner.

The Sequence Diagram for the Food Image Classification Project depicts the interactions between the user and the system during the process of classifying food images. The user initiates the process by logging into the system or registering if they are a new user. This step ensures that only authorized users can access the classification service. Upon successful login or registration, the user is granted access to the system's functionalities. This step is crucial for ensuring that the user's identity is verified and that they have the necessary permissions to use the system.

The user then collects images of food items that need to be classified. This step involves the user selecting or uploading images from their device. The collected images are pre-processed to ensure they are in the correct format and quality for accurate classification. Pre-processing steps may include resizing, normalization, and other necessary adjustments. This step is essential for improving the accuracy of the subsequent classification steps.

The pre-processed images are then segmented to isolate the food items within the images. Segmentation helps in focusing on the relevant parts of the image, which improves the accuracy of the classification process. Relevant features are extracted from the segmented images. This involves identifying and isolating the key characteristics of the food items, such as shape, texture, and colour, which are used for classification.

The extracted features are used to classify the food images using a Convolutional Neural Network (CNN). The CNN model is trained to recognize and categorize different food items, providing accurate and reliable classification results. The classification results are provided to the user. This includes the predicted food category and additional details such as nutritional information. The results are presented in a user-friendly format, allowing the user to easily understand and utilize the information.

The Sequence Diagram helps in understanding the interactions between the user and the system during the process of classifying food images. It provides a visual representation of the steps involved, from user login to the final classification results. By illustrating the key steps and their interactions, the diagram ensures that the system meets the user's requirements and provides a seamless and efficient classification process

5.3.4 ACTIVITY DIAGRAM

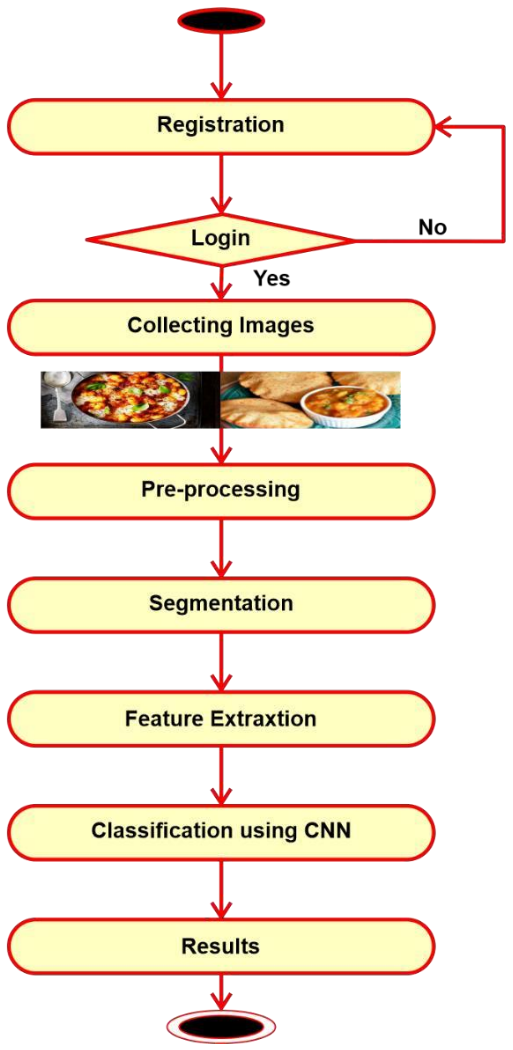


Fig 5.5 The Activity Diagram for the Food Image Classification

The Activity Diagram for the Food Image Classification Project illustrates the sequence of actions and decisions involved in classifying food images. It provides a visual representation of the steps from user registration/login to the final classification results.

Below is a detailed explanation of the Activity Diagram:

Steps and Their Descriptions

1. **Registration / Login**

   Description: The process begins with the user either registering as a new user or logging in if they already have an account. This step ensures that only authorized users can access the classification service.

   Interaction: The user is presented with options to register or log in. If the user chooses to register, they will provide the necessary details to create a new account. If the user chooses to log in, they will enter their credentials to access the system.

   Decision Point: If the user is new, they will proceed with registration. If the user already has an account, they will proceed with login.

2. **Collecting Images**

   Description: Once the user is logged in, they can start collecting images of food items that need to be classified. This step involves the user selecting or uploading images from their device.

   Interaction: The user interacts with the system to upload the food images. The system receives the images and prepares them for further processing.

3. **Pre-Processing**

   Description: The collected images are pre-processed to ensure they are in the correct format and quality for accurate classification. Pre-processing steps may include resizing, normalization, and other necessary adjustments.

   Interaction: The system applies pre-processing techniques to the uploaded images. This step is essential for improving the accuracy of the subsequent classification steps.

4. **Segmentation**

   Description: The pre-processed images are segmented to isolate the food items within the images. Segmentation helps in focusing on the relevant parts of the image, which improves the accuracy of the classification process.

   Interaction: The system performs segmentation on the pre-processed images. This step involves identifying and isolating the food items within the images, preparing them for feature extraction.

5. **Feature Extraction**

   Description: Relevant features are extracted from the segmented images. This involves identifying and isolating the key characteristics of the food items, such as shape, texture, and colour, which are used for classification.

Interaction: The system extracts features from the segmented images. This step is crucial for providing the necessary data for the classification process.

6. **Classification using CNN**

   Description: The extracted features are used to classify the food images using a Convolutional Neural Network (CNN). The CNN model is trained to recognize and categorize different food items, providing accurate and reliable classification results.

   Interaction: The system applies the CNN model to the extracted features, classifying the food images. This step involves using the trained model to predict the food category based on the extracted features.

7. **Result**

   Description: The classification results are provided to the user. This includes the predicted food category and additional details such as nutritional information. The results are presented in a user-friendly format, allowing the user to easily understand and utilize the information.

   Interaction: The system sends the classification results to the user. This step ensures that the user receives the necessary information in a clear and concise manner.

The Activity Diagram for the Food Image Classification Project depicts the sequence of actions and decisions involved in classifying food images. The process begins with the user either registering as a new user or logging in if they already have an account. This step ensures that only authorized users can access the classification service. If the user is new, they will proceed with registration. If the user already has an account, they will proceed with login.

Once the user is logged in, they can start collecting images of food items that need to be classified. This step involves the user selecting or uploading images from their device. The collected images are pre-processed to ensure they are in the correct format and quality for accurate classification. Pre-processing steps may include resizing, normalization, and other necessary adjustments. This step is essential for improving the accuracy of the subsequent classification steps. The pre-processed images are then segmented to isolate the food items within the images. Segmentation helps in focusing on the relevant parts of the image, which improves the accuracy of the classification process. Relevant features are extracted from the segmented images. This involves identifying and isolating the key characteristics of the food items, such as shape, texture, and colour, which are used for classification.

The extracted features are used to classify the food images using a Convolutional Neural Network (CNN). The CNN model is trained to recognize and categorize different food items, providing accurate and reliable classification results. The classification results are provided to the user. This includes the predicted food category and additional details such as nutritional information. The results are presented in a user-friendly format, allowing the user to easily understand and utilize the information.

The Activity Diagram helps in understanding the sequence of actions and decisions involved in classifying food images. It provides a visual representation of the steps from user registration/login to the final classification results. By illustrating the key steps and their interactions, the diagram ensures that the system meets the user's requirements and provides a seamless and efficient classification process.

6. IMPLEMENTATION

6.1 MODULES:

1. app/ml_model.py

Purpose: The core functionality for loading the trained model, making predictions, and evaluating its performance.

Key Features: The module is responsible for loading and pre-processing datasets using image transformations. This includes resizing, normalization, and other necessary pre-processing steps to prepare the dataset for inference. Additionally, it loads the modified ResNet18 model trained for specific food categories. The module provides a function to predict the class of food based on an input image, taking an image as input and returning the predicted food category. Furthermore, it evaluates the model's performance using various metrics such as classification reports and confusion matrices. This helps in understanding the model's strengths and weaknesses and provides insights into its accuracy and reliability.

2. train_model.py

Purpose: Handles the training process of the ResNet18 model.

Key Features: The module defines and applies data augmentation transformations to enhance the dataset and improve the model's ability to generalize. Techniques such as random cropping, flipping, and color jittering are used. It trains the model using the Adam optimizer and CrossEntropyLoss, iterating over the dataset and updating the model's parameters to minimize the loss function. The module validates the model after each epoch to monitor its performance on unseen data, calculating accuracy and loss metrics to track the model's progress and prevent overfitting. Finally, it saves the trained model in .pth format for future use, allowing the model to be loaded and used for inference without the need for retraining.

3. chatbot.py

Purpose: Implements a chatbot for answering user queries related to food.

Key Features: The module processes various types of user input, including questions about food origins, recipes, and nutrition. The chatbot is designed to understand and respond to a wide range of queries. It provides answers to questions specific to the dataset used for training the model, including information about the food categories, their characteristics, and any other relevant details. Additionally, it provides fullback responses for unrecognized inputs to ensure that the user always receives a relevant answer. This helps in maintaining a smooth and engaging user experience.

4. scraper.py

Purpose: Provides predefined recipes for common food items.

The module maps food items to detailed recipes, providing users with step-by-step instructions for preparing the food. This includes ingredients, cooking methods, and any special notes or tips. It gracefully handles invalid or unknown food items by providing appropriate responses. This ensures that the user is informed about the limitations of the system and guided towards valid inputs.

5. save_modified_model.py

Purpose: Modifies and saves a ResNet18 model with a new final layer adapted for the specific dataset.

The module changes the fully connected layer of the ResNet18 model to match the number of classes in the dataset. This allows the model to be fine-tuned for the specific food categories. It saves the modified model for further use, ensuring that the model can be easily loaded and used for inference without the need for retraining.

6. streamlit_app.py

Purpose: Hosts a user-friendly web interface for image classification and chatbot interaction using Streamlit.

The module allows users to upload images and get predictions with additional details like nutritional information. This provides a seamless and interactive experience for users to classify food items. It supports real-time chatbot interaction for food-related queries, allowing users to ask questions about food origins, recipes, and nutrition, and receive instant answers. The module provides an option to evaluate the model and view detailed metrics within the app, including classification reports and confusion matrices, giving users insights into the model's performance. It fetches and displays recipes on user request, allowing users to explore and learn about different food items and their preparation methods.

7. Model Training Script

Purpose: Detailed script for training the model with options to configure transformations, optimizers, and validation.

The script performs training-validation splits to evaluate the model's performance during training. This helps in monitoring the model's progress and preventing overfitting. It saves the best-performing model after training, ensuring that the most accurate and reliable model is used for inference and further evaluation.

Output: The artefacts generated include the trained model file ml_models/modified_resnet16.pth, which is saved for future use. Additionally, evaluation metrics such as classification reports and confusion matrices are generated to evaluate the model's performance and provide insights into its accuracy and reliability.

6.2 CODE SNIPPET

6.2.1. app/ml_model.py

Purpose: Core functionality for loading the trained model, making predictions, and evaluating the model.

```
# Define image transformations
train_transforms = transforms.Compose([...])
val_transforms = transforms.Compose([...])

# Load datasets
train_dataset = datasets.ImageFolder(root='data/train',
transform=train_transforms)
test_dataset = datasets.ImageFolder(root='data/test',
transform=val_transforms)

# Load the trained model
model = models.resnet18()
model.fc = torch.nn.Linear(model.fc.in_features, num_classes)
model.load_state_dict(torch.load('ml_models/modified_resnet16.pth',
map_location=torch.device('cpu')))
model.eval()

# Prediction function
def predict_image(image_path):
 image = Image.open(image_path)
 image_tensor = predict_transform(image).unsqueeze(0)
 with torch.no_grad():
 outputs = model(image_tensor)
 _, predicted = torch.max(outputs, 1)
 class_idx = predicted.item()
 class_name = train_dataset.classes[class_idx]
 nutrition = nutrition_data[nutrition_data['FoodCategory'] ==class_name]
 report = classification_report([0], [class_idx])
 matrix = confusion_matrix([0], [class_idx])
 return class_name, report, matrix, nutrition

# Evaluation function
```

```
def evaluate_model():
 model.eval()
 all_preds = []
 all_labels = []
 with torch.no_grad():
 for inputs, labels in test_loader:
 outputs = model(inputs)
 _, preds = torch.max(outputs, 1)
 all_preds.append(preds.cpu().numpy())
 all_labels.append(labels.cpu().numpy())
 all_preds = np.concatenate(all_preds)
 all_labels = np.concatenate(all_labels)
 report = classification_report(all_labels, all_preds,
target_names=train_dataset.classes)
 matrix = confusion_matrix(all_labels, all_preds)
 return all_labels, all_preds, report, matrix
```

6.2.2. train_model.py

Purpose: Trains the ResNet18 model and saves the trained model.

```
# Define image transformations
train_transforms = transforms.Compose([...])
val_transforms = transforms.Compose([...])

# Load datasets
train_dataset =
datasets.ImageFolder(root='data/train',transform=train_transforms)
test_dataset =
datasets.ImageFolder(root='data/test',transform=val_transforms)

# Load the pre-trained model
model = models.resnet18(weights=models.ResNet18_Weights.IMAGENET1K_V1)
model.fc = torch.nn.Linear(model.fc.in_features, num_classes)

# Define loss function and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)

# Train the model
def train_model(model, train_loader, val_loader, criterion, optimizer,
num_epochs=3):
 for epoch in range(num_epochs):
 model.train()
 running_loss = 0.0
 for inputs, labels in train_loader:
 inputs, labels = inputs.to(device), labels.to(device)
 optimizer.zero_grad()
 outputs = model(inputs)
 loss = criterion(outputs, labels)
 loss.backward()
 optimizer.step()
 running_loss += loss.item()
 print(f"Epoch [{epoch+1}/{num_epochs}],
Loss:{running_loss/len(train_loader):.4f}")

 # Validate the model
 model.eval()
 val_running_loss = 0.0
 val_correct = 0
 val_total = 0
 with torch.no_grad():
 for inputs, labels in val_loader:
 inputs, labels = inputs.to(device), labels.to(device)
 outputs = model(inputs)
 loss = criterion(outputs, labels)
 val_running_loss += loss.item()
 _, predicted = torch.max(outputs, 1)
 val_total += labels.size(0)
 val_correct += (predicted == labels).sum().item()
 val_accuracy = val_correct / val_total
 print(f"Validation Loss: {val_running_loss/len(val_loader):.4f},Accuracy:
{val_accuracy:.4f}")

 # Save the trained model
 torch.save(model.state_dict(), 'ml_models/modified_resnet16.pth')
 print("Trained model saved successfully.")
if __name___== '__main__':
 import multiprocessing
 multiprocessing.freeze_support()
 device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
 model = model.to(device)
 train_model(model, train_loader, val_loader, criterion, optimizer)
```

6.1.3. streamlit_app.py

Purpose: This code sets up a Streamlit app for food image classification, recipe fetching, and machine learning model evaluation.

```
import streamlit as st
from app.chatbot import generate_response
from app.scraper import get_recipe
import os
import random
from PIL import Image
from app.ml_model import predict_image, evaluate_model, train_dataset
import pandas as pd
from sklearn.metrics import classification_report

# Set up the Streamlit app
st.title("Food Image Classification App")
```

```
# Include custom CSS
st.markdown(
 """
 <style>
 @import url('static/css/style.css');
 </style>
 """,
 unsafe_allow_html=True
)

# Include custom JavaScript
st.markdown(
 """
 <script src="static/js/script.js"></script>
 """,
 unsafe_allow_html=True
)

# Upload image for prediction
uploaded_file = st.file_uploader("Choose an image...", type="jpg")
if uploaded_file is not None:

 # Save the uploaded file
 image_path = os.path.join('data/uploads', uploaded_file.name)
 with open(image_path, "wb") as f:
 f.write(uploaded_file.getbuffer())

 # Predict the image
 predicted_class, report, matrix, nutrition = predict_image(image_path)

 # Display the predicted class
 st.write(f"Predicted Class: {predicted_class}")

 # Display the image
 image = Image.open(image_path)
 st.image(image, caption=f'Uploaded Image: {uploaded_file.name}',
use_container_width=True)

 # Display the classification report
 st.write("Classification Report:")
 report_dict = classification_report([0], [0], output_dict=True,
target_names=[predicted_class])
 report_df = pd.DataFrame(report_dict).transpose()
 st.dataframe(report_df)

 # Display the confusion matrix
 st.write("Confusion Matrix:")
 st.write(matrix)

 # Display the nutritional information
 st.write("Nutritional Information:")
 st.write(nutrition)
```

```
# Input from user
st.title("CuisineChat")
user_input = st.text_input(" What's Cooking in that Brain of Yours?")
if user_input:
 if "recipe" in user_input.lower():
 food_item = user_input.lower().replace("recipe", "").strip()
 if food_item:
 st.write(f"Fetching recipe for: {food_item.capitalize()}")
 try:
 recipe = get_recipe(food_item)
 st.write(recipe)
 except Exception as e:
 st.error(f"Could not fetch recipe: {e}")
 else:
 st.warning("Please specify a food item for the recipe.")
 else:
 response = generate_response(user_input)
 st.write(response)

# Example usage to test the function
food_item = random.choice(
 ["burger", "butter_naan", "chai", "chapati", "chole bhature", "dal
makhani", "dhokla",
 "fried_rice", "idli", "jalebi", "kaathi rolls", "kadai paneer",
"kulfi", "masala dosa",
 "momos", "paani puri", "pakode", "pav bhaji", "pizza", "samosa"]
)
print(get_recipe(food_item))

# Evaluate the model
if st.button("Evaluate Model"):
 all_labels, all_preds, report, matrix = evaluate_model()
 st.write("Classification Report:")
 report_dict = classification_report(all_labels, all_preds,
output_dict=True, target_names=train_dataset.classes)
 report_df = pd.DataFrame(report_dict).transpose()
 st.dataframe(report_df)
 st.write("Confusion Matrix:")
 st.write(matrix)
```

6.2 DISCUSSION OF RESULTS

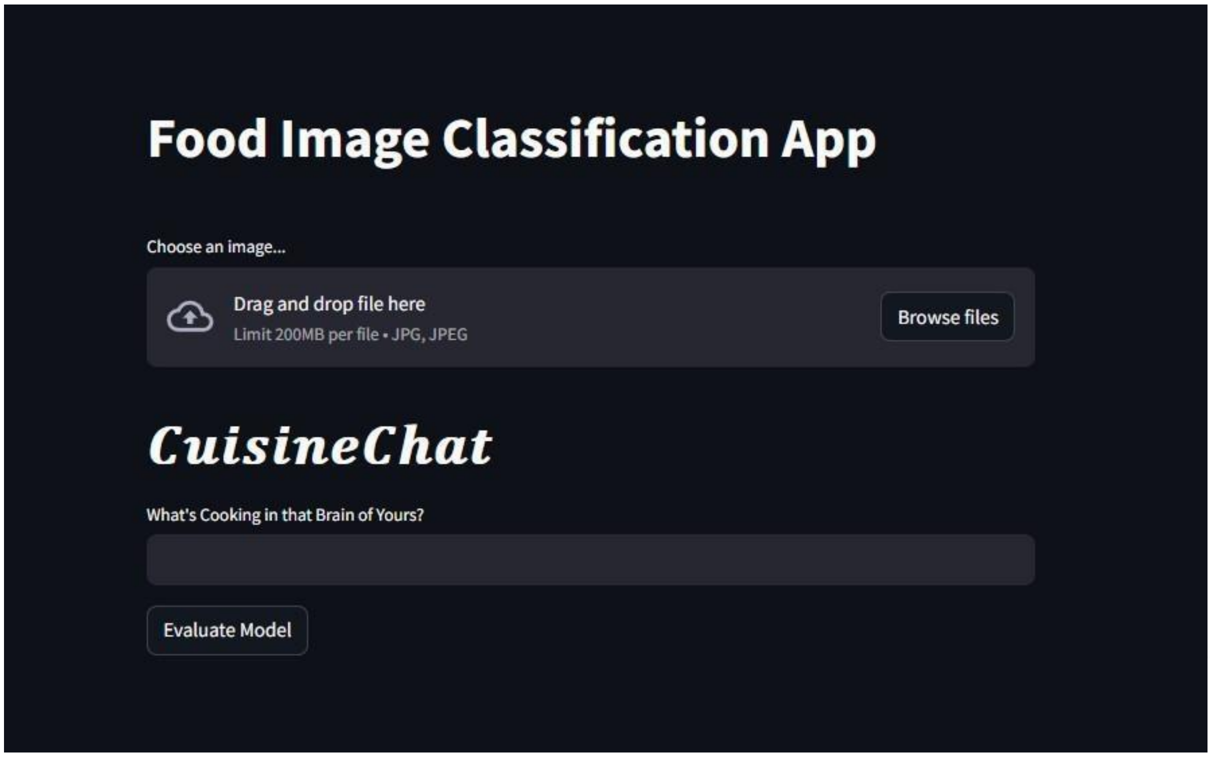
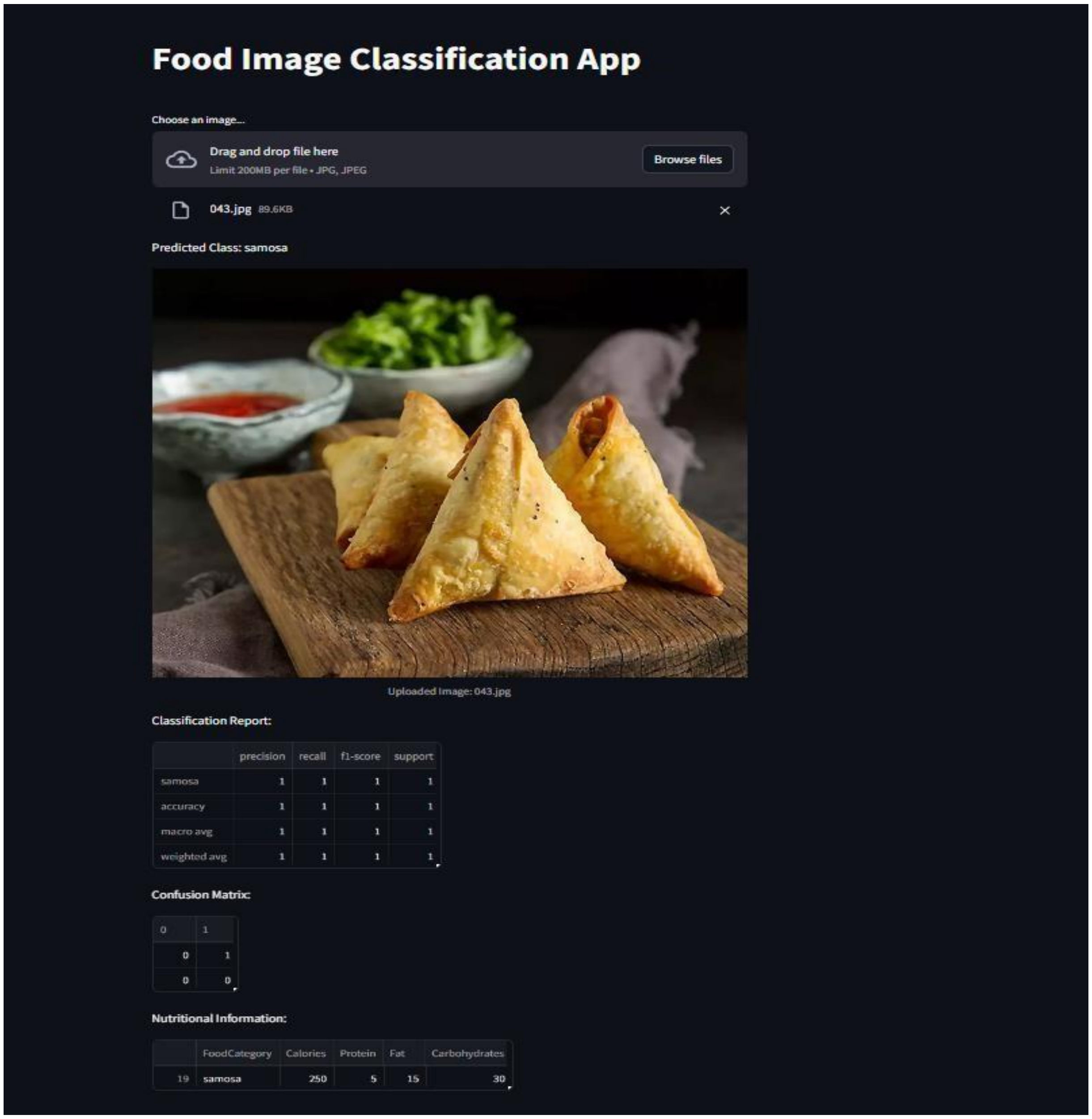

Food Image Classification App
Choose an image...
Drag and drop file here
Limit 200MB per file • JPG, JPEG
Browse files
CuisineChat
What's Cooking in that Brain of Yours?
Evaluate Model

Fig 6.1: Model Evaluation

Food Image Classification App (Initial Screen): This screenshot shows the initial interface of the Streamlit app where users can upload an image for classification.



Food Image Classification App

Choose an image...

Predicted Class: samosa

Uploaded Image: 043.jpg

Classification Report:

	precision	recall	f1-score	support
samosa	1	1	1	1
accuracy	1	1	1	1
macro avg	1	1	1	1
weighted avg	1	1	1	1

Confusion Matrix:

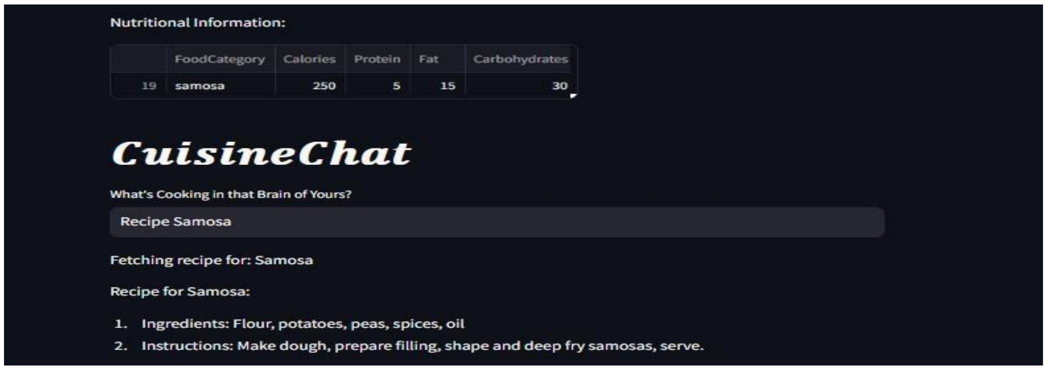
0	1
0	1
0	0

Nutritional Information:

	FoodCategory	Calories	Protein	Fat	Carbohydrates
19	samosa	250	5	15	30

Fig 6.2: Predicted Class Diagram/ Classification Report

Food Image Classification App (Prediction Result): This screenshot displays the result after an image of samosas is uploaded. The app predicts the class as "samosa" and shows the uploaded image.



Nutritional Information:

	FoodCategory	Calories	Protein	Fat	Carbohydrates
19	samosa	250	5	15	30

CuisineChat

What's Cooking in that Brain of Yours?

Recipe Samosa

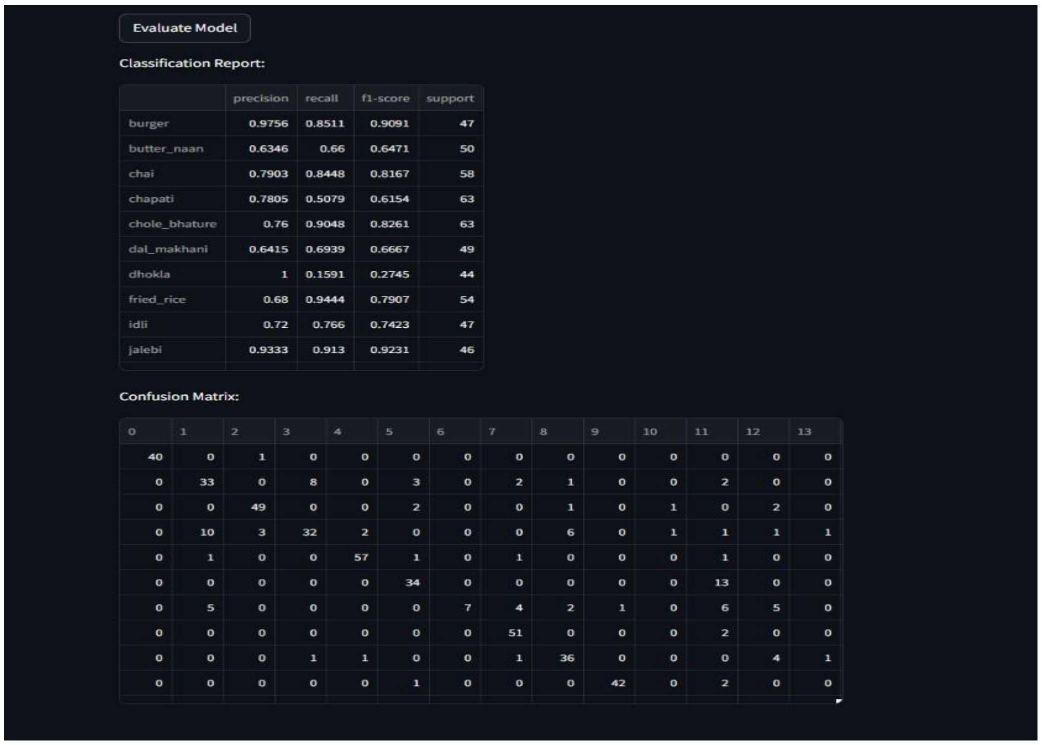
Fetching recipe for: Samosa

Recipe for Samosa:

1. Ingredients: Flour, potatoes, peas, spices, oil
2. Instructions: Make dough, prepare filling, shape and deep fry samosas, serve.

Fig 6.3: Nutritional Information

This screenshot shows the chatbot's response to a recipe request for samosas, providing a basic recipe.



Classification Report:

	precision	recall	f1-score	support
burger	0.9756	0.8511	0.9091	47
butter_naan	0.6346	0.66	0.6471	50
chai	0.7903	0.8448	0.8167	58
chapati	0.7805	0.5079	0.6154	63
chole_bhature	0.76	0.9048	0.8261	63
dal_makhani	0.6415	0.6939	0.6667	49
dhokla	1	0.1591	0.2745	44
fried_rice	0.68	0.9444	0.7907	54
idli	0.72	0.766	0.7423	47
jalebi	0.9333	0.913	0.9231	46

Confusion Matrix:

0	1	2	3	4	5	6	7	8	9	10	11	12	13
40	0	1	0	0	0	0	0	0	0	0	0	0	0
0	33	0	8	0	3	0	2	1	0	0	2	0	0
0	0	49	0	0	2	0	0	1	0	1	0	2	0
0	10	3	32	2	0	0	0	6	0	1	1	1	1
0	1	0	0	57	1	0	1	0	0	0	1	0	0
0	0	0	0	0	34	0	0	0	0	0	13	0	0
0	5	0	0	0	0	7	4	2	1	0	6	5	0
0	0	0	0	0	0	0	51	0	0	0	2	0	0
0	0	0	1	1	0	0	1	36	0	0	0	4	1
0	0	0	0	0	1	0	0	0	42	0	2	0	0

Fig 6.4: Confusion Matrix

This screenshot shows the classification report and confusion matrix resulting from the model evaluation, indicating the performance of the model across different classes.

7. TESTING

Implementation and Testing

Implementation is one of the most important tasks in a project. It is the phase in which one has to be cautious because all the efforts undertaken during the project will be very interactive. Implementation is the most crucial stage in achieving a successful system and giving users confidence that the new system is workable and effective. Each program is tested individually at the time of development using sample data and has been verified that these programs link together in the way specified in the program specification. The computer system and its environment are tested to the satisfaction of the user.

Implementation

The implementation phase is less creative than system design. It is primarily concerned with user training and file conversion. The system may require extensive user training. The initial parameters of the system should be modified as a result of programming. A simple operating procedure is provided so that the user can understand the different functions clearly and quickly. The different reports can be obtained either on the inkjet or dot matrix printer, which is available at the disposal of the user. The proposed system is very easy to implement. In general, implementation is used to mean the process of converting a new or revised system design into an operational one.

Testing

Testing is the process where the test data is prepared and used for testing the modules individually and later the validation given for the fields. Then system testing takes place, which ensures that all components of the system properly function as a unit. The test data should be chosen such that it passes through all possible conditions. Actually, testing is the state of implementation which aims at ensuring that the system works accurately and efficiently before the actual operation commences. The following is the description of the testing strategies carried out during the testing period.

7.1 System Testing

Testing has become an integral part of any system or project, especially in the field of information technology. The importance of testing is a method of justifying whether one is ready to move further, be it to check if one is capable of withstanding the rigors of a particular situation. This cannot be underplayed, and that is why testing before development is so critical. When the software is developed, before it is given to the user to use, the software must be tested to ensure it is solving the purpose for which it is developed. This testing involves various types through which one can ensure the software is reliable. The program was tested logically, and the pattern of execution of the program for a set of data was repeated. Thus, the code was exhaustively checked for all possible correct data, and the outcomes were also checked.

7.2 Module Testing

To locate errors, each module is tested individually. This enables us to detect errors and correct them without affecting any other modules. Whenever the program is not satisfying the required function, it must be corrected to get the required result. Thus, all the modules are individually tested from the bottom up, starting with the smallest and lowest modules and proceeding to the next level. Each module in the system is tested separately. For example, the image pre-processing module is tested separately. This module is tested with different images, and its approximate execution time and the result of the test are compared with the results prepared manually. The comparison shows that the results of the proposed system work efficiently compared to the existing system. Each module in the system is tested separately. In this system, the feature extraction and classification modules are tested separately, and their corresponding results are obtained, which reduces the process waiting time.

7.3 Integration Testing

After module testing, integration testing is applied. When linking the modules, there may be a chance for errors to occur; these errors are corrected by using this testing. In this system, all modules are connected and tested. The testing results are very correct. Thus, the mapping of images with features is done correctly by the system.

7.4 Acceptance Testing

When the user finds no major problems with its accuracy, the system passes through a final acceptance test. This test confirms that the system meets the original goals, objectives, and requirements established during analysis without actual execution, which eliminates the wastage of time and money. Acceptance tests are on the shoulders of users and management; it is finally acceptable and ready for operation.

Test Case

Test Case Id	Test Case Name	Expected Result	Actual Result	Test Case Status	Test Case Priority
1	Predict Food Image	Food image is identified accurately	Food image is identified accurately	Pass	High
2	Fetch Nutritional Info	Nutritional information is displayed correctly	Nutritional information is displayed correctly	Pass	High
3	Incorrect File Upload	System rejects the file with an appropriate error	System rejects the file with an appropriate error	Pass	High
4	Missing Input Handling	System prompts the user to provide valid input	System prompts the user to provide valid input	Pass	High
5	Chatbot Functionality	Chatbot provides relevant nutritional information	Chatbot provides relevant nutritional information	Pass	High

Test Cases of Food Image Classification

Test Case Id: A unique identifier for each test case.

Test Case Name: A concise description of the test case.

Expected Result: The anticipated outcome of the test case.

Actual Result: The actual outcome observed during the test execution.

Test Case Status: "Pass" or "Fail" based on whether the actual result matches the expected result.

8. CONCLUSION

In this project, The Food Image Classification project employs deep learning techniques to identify and categorize food items from images, tackling challenges like presentation variability and lighting conditions. Using CNNs and transfer learning with extensive datasets, the project aims for high accuracy and scalability. It enhances dietary monitoring and nutritional analysis, offering real-time results to support healthier eating habits. This initiative envisions a future where intelligent technology promotes a more connected and healthier global community. The food image classification project presents a robust framework for identifying food items from images and providing relevant nutritional information and recipes. The integration of a chatbot enhances user interaction, making the application more engaging and informative. However, there is significant potential for future enhancements to make the project even more comprehensive and user-friendly.

One key area for improvement is the model's accuracy. This can be achieved through data augmentation, which involves enhancing the dataset with more diverse and augmented images to improve the model's generalization. Additionally, experimenting with more advanced neural network architectures like EfficientNet or Vision Transformers, and utilizing transfer learning from models pre-trained on larger food datasets, can further boost accuracy. The user interface can also be enhanced by developing a more interactive and user-friendly dashboard with features like real-time predictions, visual analytics, and user feedback integration. Creating a mobile application version of the web app would make it easier for users to access and use the application on their smartphones.

Expanding the dataset to include a wider range of food categories, especially regional and international cuisines, would make the model more versatile. Similarly, expanding the nutritional database to cover more food items and provide detailed nutritional information would be beneficial. The chatbot can be enhanced by integrating advanced Natural Language Processing (NLP) techniques to improve its understanding and response generation. Adding support for multiple languages and implementing contextual understanding would cater to a broader audience and allow the chatbot to maintain a conversation flow.

Providing personalized recipe recommendations based on user preferences and dietary restrictions, as well as including health-focused recipes, would make the application more useful for users with specific dietary needs. Integrating the application with smart kitchen appliances and wearable devices could provide real-time cooking assistance and nutritional tracking. Adding community features like user reviews and ratings, and enabling social sharing, would create a community-driven feedback system and allow users to share their cooking experiences.

Implementing real-time analytics to track user behavior and preferences would help improve the application's features and user experience. Continuously monitoring the model's performance and updating it with new data would ensure that it maintains high accuracy. By addressing these areas, the project can evolve into a comprehensive and user-friendly tool for food recognition, nutritional guidance, and culinary exploration.

Enhancing the user interface will make the application more engaging and user-friendly. Developing a more interactive dashboard with real-time predictions, visual analytics, and user feedback integration can improve user interaction. Creating a mobile application version of the web app will allow users to access the platform conveniently on their smartphones, making it more accessible and user-centric

REFERENCES

1. Chen, M., Dhingra, K., Wu, W., Yang, L., Sukthankar, R., & Yang, J. (2009). PFID: Pittsburgh fast-food image dataset. In *2011 IEEE International Conference on Image Processing* (pp. 5413479). DOI: 10.1109/ICIP.2009.5413479

2. Kawano, Y., & Yanai, K. (2014). Food image recognition with deep convolutional features. In *Proceedings of the 2014 ACM International Conference on Multimedia* (pp. 2647866.2654970). DOI: 10.1145/2647866.2654970

3. Bossard, L., Guillaumin, M., & Van Gool, L. (2014). Indian Food Image – Mining discriminative components with random forests. In *European Conference on Computer Vision (ECCV)* (pp. 10.1007/978-3-319-10599-4_7). DOI: 10.1007/978-3-319-10599-4_7

4. Bolanos, M., & Radeva, P. (2017). Simultaneous food localization and recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (pp. 10.1109/CVPRW.2017.17). DOI: 10.1109/CVPRW.2017.17

5. Pandey, S., Shukla, S., & Chaudhary, S. (2020). Food image classification using transfer learning of deep convolution neural network. *International Journal of Advanced Research in Computer Science*.

6. Matsuda, Y., Hoashi, H., & Yanai, K. (2012). Recognition of multiple-food images by detecting candidate regions. In *2012 IEEE International Conference on Multimedia and Expo (ICME)* (pp. 10.1109/ICME.2012.103). DOI: 10.1109/ICME.2012.103

7. Tan, M., & Le, Q. (2019). EfficientNet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*. DOI: arXiv:1905.11946

8. Krishnaveni, S., Vaishnavi, T., Aravindh, S., Gowrishankar, D. J., Ashwath, K., & Madhumitha, S. Comparative Analysis of CNN-based Feature Extraction Techniques and Conventional Machine Learning Models for Quality Assessment in

Agricultural Produce. Retrieved from https://ssrn.com/abstract=5089073

9. Goh, A. M., & Yann, X. L. Food Image Classification Using Deep Learning. Retrieved from https://www.ijeea.in/wp-content/uploads/Volume-9-Issue-3-Survey-2.pdf

10. Zheng, L., Liu, G., Yan, C., & Jiang, C. Transfer Learning for Food Image Classification. Retrieved from https://researchwith.njit.edu/en/publications/improved-tradaboost-and-its-application-to-transaction-fraud-dete

11. Xuan, S., Liu, G., Li, Z., Zheng, L., Wang, S., & Jiang, C. Data Augmentation Techniques for Improved Classification. Retrieved from https://ieeexplore.ieee.org/abstract/document/8361343/figures