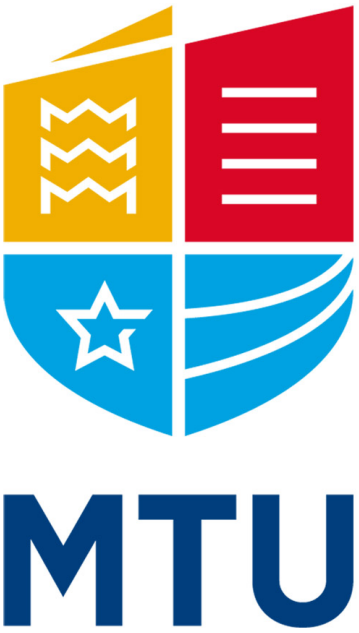


Developing an algorithm to read nutritional labels

by

Michal Strzelecki

This thesis has been submitted in partial fulfillment for the degree of Bachelor of Science in Software Development

in the
Faculty of Engineering and Science
Department of Computer Science

April 2024

Declaration of Authorship

This report, Developing an algorithm to read nutritional labels, is submitted in partial fulfillment of the requirements of Bachelor of Science in Software Development at Munster Technological University Cork. I, Michal Strzelecki, declare that this thesis titled, Developing an algorithm to read nutritional labels and the work represents substantially the result of my own work except where explicitly indicated in the text. This report may be freely copied and distributed provided the source is explicitly acknowledged. I confirm that:

- This work was done wholly or mainly while in candidature Bachelor of Science in Software Development at Munster Technological University Cork.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at Munster Technological University Cork or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this project report is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: _____

Date: _____

MUNSTER TECHNOLOGICAL UNIVERSITY CORK

Abstract

Faculty of Engineering and Science
Department of Computer Science

Bachelor of Science

by Michal Strzelecki

In today's fast-paced world, accessing accurate and clear nutritional information is paramount. This project introduces an automated system that utilises Artificial Intelligence (AI) and Optical Character Recognition (OCR) technologies to extract and interpret nutritional data from food labels. The impetus for this innovation arises from the laborious and error-prone nature of manually analysing nutritional labels. Automating this process not only enhances consumer understanding but also streamlines data management for businesses in the food industry.

The project's methodology centres around the development of an OCR application capable of efficiently processing images of food labels to extract text. Employing a combination of Python, OpenCV, and the Pytesseract OCR engine, the system achieves high accuracy in text recognition. This technology does not merely recognise characters on labels but also interprets them into a structured digital format, making it easy to integrate into various database systems. Key functional requirements include high accuracy in text extraction, resilience to variations in label designs, and compatibility with multiple database architectures, ensuring the system's applicability across different sectors within the food industry.

Acknowledgements

I would like to express my deepest gratitude to my supervisors, Oonagh O'Brien and Brian Murphy, for their invaluable guidance and unwavering support throughout the duration of this project. Their expertise and insights have been fundamental to the successful completion of my work, and their encouragement and constructive feedback have been immensely appreciated.

I must also extend my heartfelt thanks to my wife and our child, who have provided me with the motivation and inspiration needed to persevere through the challenges of this project. Their patience and understanding have been my anchor, and their love and support have fueled my commitment and passion for my studies.

To each of you, I owe a great debt of gratitude for your contributions to my academic journey and personal growth. Thank you for believing in me and standing by my side every step of the way.

Contents

List of Figures

Abbreviations

LAH	**L**ist **A**bbreviations **H**ere
AI	**A**rtificial **I**nteligence
OCR	**O**bject **C**haracter **R**ecognition

Chapter 1

Introduction

1.1 Motivation

The pursuit of a nutritious diet is more important than ever in our busy modern life. There is an obvious conflict as we approach a time where knowledge is widely available to us: the labels on food products that hold the secret to making educated dietary decisions also provide a significant difficulty. Not only is it tedious to manually translate the nutritional information from these labels, but it also creates a roadblock in our efforts to eat healthier. This project's motivation comes from a straightforward observation: nutritional information must be accessed and understood efficiently. Imagine a situation where you could just snap a picture and an intelligent application would immediately extract and arrange the data for you, saving you the trouble of reading the small print on a cereal box or the back of a canned soup. This idea is a step in the direction of encouraging better eating habits, not just a convenience.

The current method, which collects this data manually, is labor-intensive and prone to mistakes. Our goal is to solve this inefficiency by offering an automated and optimized solution. By doing this, we enable people to choose their foods with greater knowledge, improving their health and general well-being. As student of computer science, I am driven by the immense potential of Artificial Intelligence (AI), particularly in the field of Optical Character Recognition (OCR). This project is a result of strong curiosity about AI and desire to solve a practical issue. The goal is to use technology to make the process of extracting, processing, and storing nutritional information from product labels as efficient as possible.

1.2 Contribution

My decision to take on this project is primarily motivated by my professional history, educational background, and sincere interest in the domains of data analytics and artificial intelligence. These motivating elements have come together to offer me a special chance to work on a project that completely fits my interests and skill set.

- **Object-Oriented Programming:** Academically speaking, courses and modules that have given me the necessary knowledge and abilities have molded my path in computer science. My studies, especially in important modules like "Object-Oriented Programming," have given me a strong foundation in code structure. This knowledge was essential to the development of the project's software components since it guaranteed scalability and maintainability, making it possible to create a strong system that could expertly handle the extraction of nutritional information.
- **Programming for Data Analytics:** With courses like "Programming for Data Analytics," my academic research has further explored the field of data analytics. My ability to manipulate and analyze data has improved as a result of these training opportunities; these abilities are critical for preprocessing photographs and extracting nutritional information from product labels. These studies brought out my intense interest in data analytics, which has inspired me to approach this project with extreme care and attention to detail.
- **Machine Learning:** Adding to my academic background, the "Machine Learning" subject has given me a thorough understanding of AI techniques and methods. My ability to create the AI system needed to automate the extraction and parsing of nutritional data is based primarily on this understanding. In addition to being a personal interest, my love for AI is what motivates me to work hard to provide an accurate and efficient solution that makes use of machine learning concepts.

All in all, these contributions demonstrate my ability to address the particular technological difficulties this project presents. They highlight my expertise in data analytics, software design, data management, and artificial intelligence. In addition, these abilities are tools that I hope to use to creatively solve the complex problem of nutritional information extraction.

1.3 Structure of This Document

This paper is laid out to give readers an in-depth understanding of the project. There is a purpose for every chapter and section.

- **Chapter 1** an introduction to this project, it's motivations, and the structure of the document.
- **Chapter 2** guide on structuring the literature review, providing context and insights into related work.
- **Chapter 3** describes the main requirements for the problem definition, establishing a clear understanding of the project's scope.
- **Chapter 4** outlines the methodology employed, detailing the steps taken to achieve the project's objectives.
- **Chapter 5** summarizes the work done during the Implementation Phase and looks at both the achievements and the setbacks.
- **Chapter 6** dives into the discussion of the results, providing insights and potential areas for improvement.
- **Chapter 7** summarizes the project, its contributions, and future prospects.

Chapter 2

Background

The core topic of this project centers around the development of an Artificial Intelligence tool designed to automatically extract, parse, and save nutritional information from store-bought products. Within the field of Computer Science, this project falls under various core areas and main areas, positioning it as an innovative application of AI and optical character recognition technology.

2.1 Thematic Area within Computer Science

The focus area of this project combines knowledge in AI, computer vision, and data processing, and it sits at the center of multiple computer science fields. The main emphasis is on using AI and OCR technology to streamline the nutritional data extraction procedure from food packaging.

2.2 Core Areas

Within the realm of Computer Science, this project is situated in several core areas, which include:

1. **Artificial Intelligence :** The core of this project relies on AI algorithms for image recognition and text extraction. Developing an AI tool to automate the extraction and parsing of nutritional data is at the heart of this core area.

2. **Computer Vision:** Computer vision techniques are pivotal for the successful interpretation of nutritional labels from images. Algorithms for image preprocessing, character recognition, and feature extraction are essential components.

3. **Data Processing and Analysis:** Once nutritional data is extracted, it needs to be processed and stored efficiently. This involves data cleaning, transformation, and storage, aligning with this core area.

2.3 Main Areas of Computer Science

Looking more broadly, the project is under the direction of the following major branches of computer science:

1. **Software Development:** Developing an AI tool that automatically extracts information from images necessitates robust software development skills. Python, along with libraries like Numpy, Scikit-Learn, Scipy, and Pandas, is a crucial part of this endeavor.

2. **Machine Learning:** AI is a subset of Machine Learning, and this project leverages machine learning techniques for character recognition, text extraction, and accuracy assessment.

3. **Data Science:** The project involves a significant data science component, as it entails data preprocessing, feature engineering, and data analysis to ensure the extracted nutritional information is accurate and reliable.

It is clear from placing this project inside these fundamental and important areas of computer science that it handles a challenging issue at the intersection of artificial intelligence, computer vision, and data processing. Simplifying the procedure for gathering and analyzing nutritional information from food packaging is the aim, which will ultimately support the larger campaign to encourage wholesome eating and living.

2.4 A Review of Optical Character Recognition

With the aid of optical character recognition (OCR) technology, text images can be transformed into machine-readable text that can be modified, searched, or handled by other programs. OCR can be applied to a number of tasks, including text translation, data extraction, document digitization, and the creation of accessible material. OCR is a branch of artificial intelligence and computer vision that uses a variety of methods, including machine learning, natural language processing, pattern recognition, and image processing. OCR presents a variety of methods, each suited to a certain set of requirements. Google maintains Tesseract, an innovative open-source OCR technology that is flexible enough to handle a wide range of document types and has extensive language support, however some applications may need to be customized. In the business world,

Abbyy OCR stands out for its exceptional accuracy and capacity to handle complicated documents however at a premium price. While its reliance on cloud connection may be a negative for offline apps, Google Cloud Vision excels at text recognition across several languages and combines cloud-based ease with deep learning techniques, making it suitable for integrated digital environments. Paddle OCR and EasyOCR, both free and open-source, stand out for their language versatility and ease of use, making them accessible choices for a wide range of users. Keras OCR, leveraging the power of the Keras deep learning library, offers a balance of functionality and user-friendly design but may fall short in accuracy compared to its counterparts. In sum, the selection of an OCR tool hinges on a delicate balance between accuracy, complexity, language support, usability, and budget, underscoring the importance of aligning the tool with the specific requirements of the task at hand[2][3].

We will examine the primary approaches, strategies, and models utilized in OCR in this review and weigh their advantages and disadvantages. We will also talk about OCR's upcoming trends and directions as well as its present difficulties and constraints. The three primary stages of OCR are text recognition, text detection, and imagine processing.[4] Every stage uses a variety of models, strategies, and procedures to increase OCR's precision and effectiveness. The initial stage of OCR is image processing, which involves enhancing the input image's quality and readability. Several widely used methods for image processing include:

- Binarization: This is the process of converting a grayscale or color image into a black-and-white image, where each pixel is either 0 (black) or 1 (white). Binarization helps to reduce the noise and complexity of the image, and to highlight the text regions. There are various algorithms for binarization, such as Otsu's method, Sauvola's method, Niblack's method, etc.[5]
- Skew correction: This is the process of correcting the orientation and alignment of the text in the image. Skew correction helps to improve the readability and segmentation of the text. There are various algorithms for skew correction, such as Hough transform, projection profile, Radon transform, etc.[6]
- Noise removal: This is the process of removing unwanted pixels or regions from the image, such as speckles, blobs, holes, etc. Noise removal helps to improve the clarity and quality of the image. There are various algorithms for noise removal, such as median filter, Gaussian filter, morphological operations, etc.[7]

Text detection can be categorized into two types: scene text detection and document text detection. Finding text in natural situations, such street signs, billboards, or product labels, is known as scene text detection[8]. The process of document text detection

involves finding text in scanned documents including receipts, invoices, and forms. Since scene text might have different fonts, sizes, colors, orientations, backdrops, and distortions, scene text identification is more difficult than document text detection. Several popular approaches and strategies for text recognition include:

- Edge-based methods: These methods use edge detection algorithms, such as Canny, Sobel, or Laplacian, to find the boundaries of text regions. Edge-based methods are fast and simple, but they are sensitive to noise and low contrast, and they cannot handle complex text layouts or curved text.[9]
- Texture-based methods: These methods use texture analysis algorithms, such as Gabor filters, local binary patterns, or wavelets, to extract textural features from text regions. Texture-based methods can handle various fonts and colors, but they require training and classification, and they cannot handle complex text layouts or curved text[10].
- Machine learning-based methods: These methods use machine learning algorithms, such as support vector machines, random forests, or neural networks, to learn text features from labeled data and classify text regions. Machine learning-based methods can handle various fonts, colors, orientations, backgrounds, and distortions, but they require large and diverse datasets, and they are computationally expensive[11].
- Deep learning-based methods: These methods use deep neural networks, such as convolutional neural networks, recurrent neural networks, or attention mechanisms, to learn text features from large and unlabeled data and detect text regions. Deep learning-based methods can handle complex text layouts, curved text, overlapping text, and touching text, and they achieve state-of-the-art performance, but they require high-end hardware and optimization[12].

Some of the popular models and frameworks used for text detection are:

- Efficient and Accurate Scene Text Detector: EAST is a fast and accurate model for scene text detection, based on a fully convolutional network that directly predicts the text regions and their orientations. EAST can handle various text orientations and scales, and it can run in real time on GPU.[13]
- Convolutional Recurrent Neural Network: CRNN is a powerful model for text recognition, combining the benefits of CNNs and Recurrent Neural Networks (RNNs). This architecture is adept at handling sequential data, making it ideal for recognizing text in various conditions, including challenging environments with distorted or poorly aligned text[14].

- Tesseract: This is an open source OCR engine for document text detection and recognition, based on a hybrid approach that combines traditional methods and machine learning methods. Tesseract can handle various languages and fonts, and it can be integrated with other applications.[15]
- OpenCV: This is an open source computer vision library that provides various functions and modules for text detection and recognition. OpenCV can handle various image formats and sources, and it can be used with Python, C++, or Java.[16]

Text recognition is the process of converting the extracted text regions into machine-readable text. Optical character recognition and optical word recognition (OWR) are the two categories under which text recognition falls. While OWR is responsible for identifying entire words from text regions, OCR is responsible for identifying individual letters from text regions. While OWR is more reliable and efficient than OCR, OCR is more precise and fine-grained[17]. Several popular approaches and strategies for text recognition include:

- Template matching methods: These methods use predefined templates of characters or words, and compare them with the text regions using similarity measures, such as correlation, distance, or shape. Template matching methods are simple and fast, but they are sensitive to noise, rotation, scaling, and deformation, and they cannot handle unknown or variable fonts[18].
- Feature extraction methods: These methods use feature extraction algorithms, such as histogram of oriented gradients, scale-invariant feature transform, or local feature descriptors, to extract distinctive features from text regions, and match them with the features of characters or words. Feature extraction methods are robust to noise, rotation, scaling, and deformation, but they require training and classification, and they cannot handle complex or cursive fonts[19].
- Machine learning methods in OCR, such as support vector machines, hidden Markov models, and neural networks, begin with feature extraction, where key attributes of text are identified from labeled data. In the learning phase, these algorithms use the extracted features to recognize patterns, enabling them to identify text in various fonts and styles, even if they are unknown. Although powerful in processing diverse text data, these methods require large datasets and are computationally intensive due to the complexity of feature analysis[20].
- Deep learning methods: These methods use deep neural networks, such as convolutional neural networks, recurrent neural networks, or attention mechanisms, to

learn the features and patterns of characters or words from large and unlabeled data, and recognize them from text regions. Deep learning methods can handle complex or cursive fonts, and they achieve state-of-the-art performance, but they require high-end hardware and optimization[21].

Some of the popular models and frameworks used for text recognition are:

- CRNN: This is a convolutional recurrent neural network model for scene text recognition, based on a combination of convolutional layers, recurrent layers, and a connectionist temporal classification layer. CRNN can handle variable-length text and various fonts, and it can be trained end-to-end[22].
- Attention OCR: This is an attention-based neural network model for scene text recognition, based on a combination of convolutional layers, recurrent layers, and an attention mechanism. Attention OCR can handle variable-length text and various fonts, and it can learn the alignment between the text regions and the output text[23].
- Tesseract is an open-source OCR engine designed for document text detection and recognition, utilizing a hybrid approach that melds traditional techniques with machine learning methods. Capable of handling a diverse array of languages and fonts, Tesseract can be integrated into various applications. PyTesseract, a Python wrapper for Tesseract, extends its functionality, enabling seamless use with Python and OpenCV. This wrapper supports a range of image formats and sources and offers customization through various parameters and options, enhancing Tesseract's versatility in different OCR tasks[24].

OCR is an effective and useful technology, but it has a number of problems and restrictions that affect how well it works. These barriers include the input image quality, where errors in word recognition can be introduced by elements like as noise, blur, low resolution, contrast, and illumination[25][26]. Image quality can be improved by preprocessing methods such normalization, denoising, enhancement, and binarization[27][28]. Additionally, segmenting and ranking text sections becomes challenging in complicated text layouts with several columns, tables, pictures, and mixed languages. This calls for the use of postprocessing techniques like.

- Layout Analysis: This method involves analysing a document's spatial arrangement to determine its organisational structure. Different elements, such as text blocks, graphics, tables, and headers, are recognised and segmented. This is essential for reliably extracting and recreating data from complex texts with a variety of content types[29].

- Document Structure Analysis: This process focuses on understanding the hierarchical structure of a document, such as headings, paragraphs, lists, and footnotes. It helps to identify the logical flow of information in the document which is essential for tasks like document summarising and content categorisation[30].
- Language Identification: This method is used to identify the language used in a document's text. It is especially important for multilingual documents as it directs the OCR process to use suitable character sets and linguistic conventions for precise text extraction and recognition[31].

Advanced approaches including adaptive thresholding, perspective correction, and font identification are used to overcome issues in matching and categorizing text that arise from variations in text appearance, such as different fonts, sizes, colors, orientations, backgrounds, and distortions. Moreover, unclear text meaning caused by misspellings, acronyms, symbols, or abbreviations can impair understanding and introduce mistakes or meaningless words, which can be avoided by using semantic strategies like word extension, spell checking, and symbol identification.Although OCR technology has completely changed how we handle textual data, there are still a number of issues and restrictions that limit its usefulness. An important obstacle is that OCR accuracy quickly deteriorates with low-quality images, which includes problems like noise, blur, and low resolution. Complicated document layouts with a combination of text, graphics, and tables present more difficulties and frequently result in inaccurate text extraction or loss of structural data. Because handwriting styles vary greatly from one another and because there is a lack of standards, handwriting recognition is still an extremely difficult task. OCR systems also have trouble with typeface and language diversity, especially when processing texts with odd fonts or non-Latin scripts. The absence of contextual and semantic knowledge, which is crucial in applications like legal document analysis and summary, is another significant constraint. Lastly, as present systems frequently require extensive manual involvement to repair mistakes, the requirement for effective error correction and post-processing techniques is important to improve OCR accuracy[32][33].

OCR is expected to continue growing and evolving quickly in the future, with a number of interesting new paths and trends emerging. To improve OCR performance and accuracy, deep learning, a subset of machine learning, is leading the way[34][35][36]. It uses deep neural networks to understand text with complicated layouts, curved text, overlapping text, and touching text. It also masters the alignment between text regions and output text. Moreover, deep learning can work in concert with other methods such as reinforcement learning, generative adversarial networks, and attention mechanisms. Multimodal OCR is an intriguing approach that leverages several modalities such as text, audio, video, and photos to enhance OCR outcomes by logically combining diverse

yet complimentary data. Multimodal OCR demonstrates its efficacy in processing complex, multimodal documents, including books, magazines, and comics that contain text, images, sounds, and animations. Its capabilities also extend to document translation, retrieval, and summary. Last but not least, Semantic OCR has become a prominent trend, utilizing semantic data that includes knowledge, context, and logic to correct mistakes, clear up confusion, and enhance textual meanings. This method opens up new possibilities for document analysis, comprehension, and generation by not only comprehending the structure, content, and purpose of documents but also producing semantic representations.

Chapter 3

Developing an algorithm to read nutritional labels

3.1 Problem Definition

In order to improve the challenges related to the human extraction and interpretation of nutritional data from food product labels, the "Automated Nutritional Information Extraction System" project was created. The main issue is that nutritional information processing and interpretation take a lot of time and are prone to errors, which makes data difficult for customers to obtain and understand.

This project's main goal is to create an AI-powered software program that uses optical character recognition (OCR) technology to automatically extract, process, and integrate nutritional data from a range of food product labels. With the help of this software, businesses in the food sector can easily extract and incorporate nutritional data into their current databases.

The project's possible risks are mostly related to the reliance on outside APIs for data parsing and image recognition. The smooth running of the project may be hampered by issues or modifications to the API's features.

The project's goal is to provide a reliable, accurate, and effective solution that makes it simple for businesses to compile dietary data into their databases. The core issue this project aims to solve is captured by the software's emphasis on accuracy, efficiency, and compatibility with current databases.

3.2 Objectives

The "Automated Nutritional Information Extraction Software" project's goals are designed to meet the specific needs for a more efficient method of obtaining and incorporating nutritional information from food product labels into databases. These objectives serve as guiding milestones for the project's progress:

1. Dataset Acquisition: Gather comprehensive dataset containing diverse range of food and product labels.
2. Data Cleaning and Labelling: Clean and process the dataset by removing noise, standardising formats and correcting inconsistencies
3. Model Exploration:
   (a) Research and evaluate existing OCR models suitable for text extraction from images
   (b) Consider pre-trained models such as Tesseract, CRNN or EAST.
4. Data Preparation for Training: Preprocess the labeled dataset into a format suitable for morel training, ensuring compatibility with chosen model architecture.
5. Model Training Train selected model using prepared dataset to recognize and extract nutritional information accurately
6. Hyperparameter Adjustment: Adjust model hyperparameters to optimise performance.
7. Validation of Dataset Creation: Set aside a portion of dataset as validation set to verify model's performance.
8. Model Evaluation Evaluate the trained models performance using the validation set to measure accuracy, precision, and any other test scores.
9. Model Refinement: Iterate on the model based on the evaluation results and make adjustments to improve accuracy
10. Testing Test refined model on new data to ensure consistent and accurate extraction across diverse label formats and conditions.
11. Compatibility with Various Databases and Layouts: Ensure seamless integration with a wide array of database architectures and formats commonly used in food industry, adapting to different layouts and structures for efficient data integration

12. Scalability and Performance Optimisation: Develop the program with scalability in mind to ensure that large volumes of label data are processed effectively. Enhance performance to provide dependable and quick extraction, even with larger data loads.

13. Comprehensive Documentation and Support: Provide complete instructions on how to integrate the software with current systems, together with supporting documentation, to ensure a smooth implementation process. Keep resources for assistance and troubleshooting easily available.

3.3 Functional Requirements

1. Image Processing and Text Recognition

   (a) Image Enhancement: Develop algorithms for optimal image quality.

   (b) Text Detection: Implement accurate text region localization.

   (c) OCR Engine: Create precise nutritional data extraction.

2. Accuracy and Validation

   (a) Data Validation: Verify extracted data accuracy.

   (b) Error Handling: Identify and rectify extraction errors.

3. Performance Optimization

   (a) Processing Efficiency: Ensure swift label image processing.

   (b) Scalability: Design for handling increased data volumes.

4. User Documentation

   (a) Extraction Module Guide: Detailed usage documentation.

5. API Integration

   (a) API Adaptability: Efficiently integrate and manage external APIs.

6. Customization

   (a) Configurable Settings: Adapt to diverse label formats.

3.4 Non-Functional Requirements

1. Performance
   (a) Fast: Ensure rapid image processing and data extraction.
   (b) Reliability: Maintain consistent and accurate extraction performance.
2. Scalability
   (a) Capacity: Scale system efficiently with increasing data volumes.
   (b) Flexibility: Adapt to varying database sizes and complexities.
3. Accuracy and Error Handling
   (a) Precision: Achieve high accuracy in data extraction.
   (b) Error Resilience: Implement robust error detection and correction mechanisms.
4. Usability
   (a) User-Friendly: Provide an intuitive and easy-to-use system.
   (b) Compatibility: Ensure compatibility across diverse platforms and devices.

Chapter 4

Implementation Approach

4.1 Architecture

The OCR application for reading nutritional labels will have a simple architecture that concentrates on local processing and storage without a complicated user interface. This architecture is perfect for usage as a backend or for direct connection with current systems.

4.1.1 Core Components:

- OCR Module: Text recognition is powered by Pytesseract, a Python wrapper for Google's Tesseract-OCR Engine. It will be used to interpret textual information from nutritional label images. To improve the accuracy of OCR, OpenCV (Open Source Computer Vision Library) will be utilised for initial image processing tasks like loading, converting, and pre-processing (noise reduction, thresholding, etc.).
- Data Processing: The OCR extracted data will be cleaned, standardized, and formatted by a Python based package. Accuracy will be ensured by doing data validation and processing the unstructured text into a structured format like CSV or JSON.
- Local Storage: A local database or file system will hold the processed data. SQLite or even a flat-file system like CSV or JSON files would be sufficient for storing and retrieving the processed nutritional information given the project's scalability and simplicity.
- User Interface: For simple tasks like starting the OCR process, setting image paths, and retrieving the processed data, a command-line interface may be incorporated.

The backend-focused design and simplicity of the project are well suited to this CLI methodology.

4.1.2 Workflow

1. Image Input and Processing:
    - This stage involves loading nutritional label images using OpenCV, followed by preprocessing steps like grayscale conversion, noise reduction, and thresholding to optimize them for OCR.
2. Text Recognition with Pytesseract:
    - Preprocessed images are then fed into Pytesseract for text extraction, focusing on accurately capturing textual data from the labels.
3. Data Interpretation and Structuring:
    - The extracted raw text undergoes processing to structure it into a readable format, involving parsing techniques and data validation to ensure accuracy and consistency.
4. Local Data Storage:
    - Structured data is stored locally in a database or file system, such as SQLite, CSV, or JSON, chosen for simplicity and scalability.
5. Command Line Interface:
    - A CLI is potentially incorporated for basic operations like initiating the OCR process and retrieving processed data, enhancing user interaction with the backend-focused system.

4.1.3 Hardware and Software Requirements

- A standard computer or laptop setup with enough processing power to run Python, OpenCV, and Tesseract.
- Software requirements include Python, OpenCV, Pytesseract, and necessary libraries for data handling and storage.

4.2 Risk Assessment

1. Identifying Risks:

    - OCR Accuracy Dependency: The main concern is Pytesseract's ability to accurately interpret text from photos. Changes in typefaces, picture quality, and label designs have a big influence on OCR accuracy.
    - Data Processing Errors: The data processing module carries some risks, such as improper OCR output formatting and parsing.
    - Local Storage Limitations: Potential restrictions on local storage systems would be scalability and mainly accessibility.

2. Risks Mitigation:

    - Enchance OCR Accuracy: By using OpenCV to apply advanced image preprocessing methods to enhance picture quality prior to OCR processing. Pytesseract settings should be updated and calibrated on a regular basis to accommodate various label formats.
    - Robust Data Processing: To reduce processing mistakes strict data validation and error checking methods should be developed. To guarantee the dependability of the data processing module, there is constant testing using a variety of label samples.
    - Addressing Storage Limitations: Evaluating the selected local storage solution's performance and storage capacity. Investigating choices for simple data recovery and backup.

4.3 Methodology

The approach used for the project is a combination of agile project management, incremental development, and extensive research, meant to address the challenges involved in creating an OCR-based system that will extract nutritional information from food labels. The project begins with a thorough analysis of the literature on OCR technologies, with a particular emphasis on how these technologies are used to extract text from pictures, particularly nutritional labels. This stage involves studying academic papers, industry reports, and case studies to understand the current state of the art. Examining Pytesseract and OpenCV's technical features and constraints will take up a large portion of the research. Furthermore, a comparative evaluation of several OCR tools and image preprocessing methods will be carried out to determine the best set of parameters for text extraction from food labels.

The development process is represented by gradual construction and prototyping. Prototypes in the early stages will test several OCR settings and preprocessing techniques, enabling prompt feedback and modifications. Quick testing and improvement are made possible by this iterative procedure. The development process will be step-by-step, beginning with fundamental features and working up to more intricate ones. Feedback loops will be essential, with tweaks made in response to test findings and frequent testing using sample photographs. In order to guarantee that the system meets real world requirements and usability standards, feedback from potential users. The base of project management will be an agile methodology which promotes flexibility and iterative development. The project will be divided into smaller, more doable jobs, each with a distinct due date. I will track tasks and monitor progress using tools like Trello boards and Gantt charts. Consistent mentor meetings and evaluations will enable ongoing progress evaluation and adjustment. The process of detecting, evaluating, and reducing any risks such as technological difficulties and project schedule delays will be continuous.

The project will place a strong emphasis on individual skill enhancement in Python, Pytesseract, and OpenCV. Self directed study using online resources, tutorials, and courses designed especially for modern technologies will be the method of instruction for this approach. Considering this is a solo project, it might be valuable to participate in online groups and forums centered around OCR and image processing. These platforms could be a valuable resource for troubleshooting, learning, and remaining current with industry innovations. The ability to use and hone these skills will be crucial, and every stage of the project will offer opportunities to do so. Guidance will be given by regular meetings with my mentor or supervisor, who will make sure the project stays on course and any technical issues are quickly resolved. To ensure the effectiveness and dependability of the system, strict procedures will be used during the assessment and testing phase. Continuous testing will be used to evaluate the overall performance of the system as well as unit test individual components during the development cycle. To assess the efficacy of the system quantitatively, key performance metrics including accuracy, processing speed, and system dependability will be developed. Because this is a solo project, it will be necessary to develop a systematic testing strategy that is manageable by one person. During this stage, input from my mentor or supervisor will be quite important as it will offer an outside viewpoint on the system's functionality and point out areas that need work. The OCR system will be refined through iterative testing and feedback loops to ensure that it satisfies the defined objectives and performs well in real world circumstances.

- How to tackle the needed research to fulfill the background chapter.

- How to set up your Computer Science skills to the project needs (e.g., describe your plan to learn any new technology involved on the project that you are not familiar with).
- What core project managing approach will you follow (e.g., Waterfall, Scrum, etc).

4.4 Implementation Plan Schedule

Week 1: Project Initiation

- Set up the development environment.
- Begin a literature review on OCR technologies, focusing on Pytesseract and OpenCV.

Week 2: Initial Exploration and Prototyping

- Familiarize with Pytesseract and OpenCV.
- Develop a basic OCR prototype with sample images.

Week 3-4: Advanced Image Preprocessing

- Implement and refine image preprocessing techniques.
- Test OCR accuracy with enhanced preprocessing methods.

Week 5-6: OCR Optimization and Data Structuring

- Fine-tune Pytesseract settings for improved accuracy.
- Develop scripts for parsing and structuring OCR output.

Week 7-8: Local Storage and Intermediate Testing

- Implement local data storage solution.
- Conduct extensive testing with diverse images.
- Start integrating feedback and adjustments.

Week 9: System Integration and CLI Development

- Begin integrating OCR, preprocessing, and storage components.
- Develop a basic command-line interface for system interaction.

Week 10: Comprehensive System Testing

- Perform thorough integration testing of the complete system.
- Refine system components based on test results.

Week 11: Final Refinements and Documentation

- Make final adjustments and optimizations.
- Start preparing detailed documentation and user instructions.

Week 12: Project Finalization and Review

- Finalize the project, ensuring all components work seamlessly.
- Review and reflect on the project with your supervisor/mentor.
- Prepare for project submission and presentation.

4.5 Evaluation

My goal in evaluating my project is to conduct a comprehensive and broad analysis. The approach starts with the methodical gathering and examination of a wide range of nutritional label photos. My goal is to make sure that the accuracy and functionality of the OCR system are thoroughly tested on a variety of label types. A large portion of my work is comparing the OCR system's primary features to the original project objectives and evaluating how well it can meet the criteria for a minimal viable product (MVP). To determine if the system is prepared for operational deployment, this assessment is necessary.

I intend to analyse any other features or improvements I have added to the project in addition to the basic features study. This involves evaluating how they add to the system's overall usefulness and worth. The analysis of the initial issue that the project was meant to solve is an important component of this review. In order to keep the project on track with its main goal, it is imperative that this reflection be used to assess if the

produced system successfully satisfies the stated objectives and addresses the intended problem.

My complete assessment approach takes into account technical proficiency, adherence to the original project objectives, and the efficiency of problem-solving techniques. By taking into account all of these different factors, I hope to assess the system's technological abilities as well as its usefulness and relevance to the problems it was created to solve. This strategy is crucial because, considering that this project is being completed alone, it guarantees that every aspect of the work is examined for effectiveness and efficiency. The end result is a reliable and efficient OCR system that serves the intended goal.1

4.6 Prototype

The application prototype is described in Chapter 4 as a high-level, simplified system that efficiently converts image input to text output. The image input is the first step in the prototype. To improve image quality and get it ready for text detection, OpenCV is used for pre processing. After that, the Pytesseract OCR engine takes over to identify and locate text within the picture. After recognition, the text is reorganised into a format that can be used, like CSV or JSON. A clear, structured text that is prepared for database integration or additional analysis is the result. This prototype, while basic, captures the essential functions expected of the final system, following to the simplicity and back-end focused nature of the project without additional complexities.

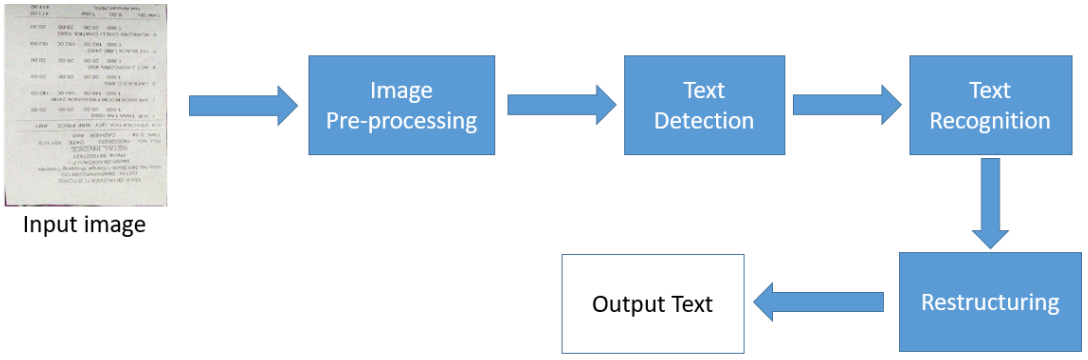


FIGURE 4.1: Basic OCR Pipeline[1]

Chapter 5

Implementation

The initial phase of my project explored adapting an existing OCR technology to extract nutritional information from food labels. While this approach utilised established OCR frameworks such as Google's Tesseract, it quickly became evident that adapting OCR to accurately capture diverse and complex food label formats was both technically challenging and time-consuming.

1. Challenges with OCR Adaptation: Food labels exhibit a wide variety of image quality, designs, fonts, and layouts that significantly complicate the text extraction process. The customisation required for OCR to handle such variability would necessitate extensive training and fine-tuning. The critical nature of nutritional information demands extremely high accuracy, which the adapted OCR might not consistently achieve without substantial iterative improvements and validations.

2. Transition to a Comprehensive Mobile Solution: Considering these insights, the project was reoriented towards the development of a mobile application that incorporates barcode scanning, PyTorch clustering, and Flutter development, aligned with the Open Food Facts API. This approach better demonstrates a holistic use of technologies and provides a multifaceted educational tool:

    - Barcode scanning offered a straightforward and reliable method to identify products and retrieve data. This approach bypasses the complexities associated with OCR by directly accessing the Open Food Facts API, which provides extensive, accurate nutritional data
    - Choosing Flutter for app development allowed for a seamless, cross-platform solution that can reach a broader audience on both Android and iOS devices. Flutter facilitates a faster development cycle with its hot reload features and

a rich set of pre-designed widgets that enhance the user interface and experience. Potential restrictions on local storage systems would be scalability and mainly accessibility.

- Integrating a PyTorch clustering model enabled the application to analyse nutritional data intelligently. It classifies products into health-related clusters, providing users with immediate recommendations about the nutritional quality of their food choices, which adds a layer of personalized insight to the app.

The strategic pivot from a specialised OCR adaptation to a more integrated mobile application was driven by a desire to produce a more demonstratively rich project. It reflects a better understanding of applying multiple technologies in harmony to solve practical problems and provides a platform to effectively showcase these skills to lecturers, students, and potential employers during my final year presentation.

5.1 Difficulties Encountered

Easy Difficulty

Title: Database Initialization and Basic CRUD Operations
Affected Area: Mobile Application Database Management

Description of Difficulty:
The project required setting up a local SQLite database to store product data. Initially, defining the schema and ensuring that all necessary fields were included for the product model was straightforward.

Solution:
The database was successfully initialized using the `DatabaseHelper` class, which provided methods for creating, reading, updating, and deleting product entries. This setup allowed for smooth operations without significant issues, leveraging SQLite's capabilities within a Flutter environment.

Title: Data Cleaning and Preprocessing
Affected Area: Data Preparation

Description of Difficulty:
The initial stages of data cleaning, such as removing unnecessary columns and filtering out rows with extreme values, were straightforward. The script handled these tasks

efficiently by dropping irrelevant columns and ensuring that nutritional values were within a reasonable range.

Solution:
Utilizing pandas for data manipulation, the script was able to clean the dataset by removing columns and rows that did not contribute to the analysis. This step was essential for preparing the data for further processing and did not pose significant challenges.

Medium Difficulty

Title: Integration of PyTorch Model with Flutter Application
Affected Area: Mobile Application Machine Learning Model Integration

Description of Difficulty:
Integrating the PyTorch machine learning model into the Flutter app posed challenges, particularly in managing the data flow between the Flutter frontend and the native backend using `MethodChannel`. The complexity of serializing the product data, sending it across the platform channel, and receiving the classification results needed careful handling.

Solution:
The integration was achieved by setting up a `MethodChannel` that facilitated communication between Dart and Kotlin code. This setup involved serializing the product data into a format that could be sent over the channel, and deserializing it on the native side to be fed into the PyTorch model. The classification results were then sent back to the Flutter app to be displayed to the user.

Title: Principal Component Analysis (PCA) and K-Means Clustering
Affected Area: Data Analysis and Clustering

Description of Difficulty:
Performing PCA and K-Means clustering to categorize products based on their nutritional content presented a moderate challenge. The script had to determine the optimal number of components for PCA and then apply K-Means clustering to the reduced dataset.

Solution:
The script dynamically selected the number of PCA components by retaining those that contributed to a specified variance threshold. After PCA reduction, K-Means clustering was applied to segment the products into distinct groups. The script also included

visualization functions to aid in interpreting the results, such as plotting PCA loadings and cluster distributions.

Hard Difficulty

Title: Persistent Storage of Clustering Results
Affected Area: Mobile Application Data Management

Description of Difficulty:
After implementing the clustering algorithm to categorize products based on their nutritional content, the next challenge was to persist these results within the app's local database. The goal was to save the clustering results so that users could revisit their product history without needing to reprocess the data. However, difficulties arose in modifying the database schema to include these results effectively.

Solution:
A temporary workaround was implemented due to the complexities involved. Instead of persisting the clustering results directly in the database, an action button was added within the product history view. This button allows users to reinitiate the clustering process for individual products. While this solution does not offer the optimal user experience of instant data retrieval, it ensures that users can still access updated clustering information on demand.

Title: Conversion of Scikit-Learn Models to PyTorch for Mobile Integration
Affected Area: Model Deployment

Description of Difficulty:
The most challenging part was converting the trained Scikit-Learn models (PCA and K-Means) to a format compatible with PyTorch for deployment in a mobile application. The script had to ensure that the transformation and clustering operations performed by the original models were accurately replicated in PyTorch.

Solution:
Custom PyTorch modules were created to mimic the behavior of the original Scikit-Learn models. These included a custom scaler, PCA, and K-Means classes, which were then combined into a single pipeline model. The model was serialized and saved for use in the mobile application, ensuring that the clustering functionality could be utilized on-device.

Enumerate the different difficulties you have found when developing your solution approach. Create three categories of difficulties:

- **Easy**: You managed to solve the problem with little difficulty.
- **Medium**: It was not easy to solve, but you managed to develop a workaround or solution and still achieve the functionality you originally had in mind.
- **Hard**: The difficulty was so complicated that you didn't managed to solve it. As a result, some functional requirement / non-functional requirement or use case from your solution approach was not achieved.

5.2 Actual Solution Approach

From January to April, significant development and adaptations were made to my project. These changes were necessitated by various challenges encountered during the implementation phase, as detailed in Section 5.1. Below, I compare the original design and the final project across the components initially outlined:

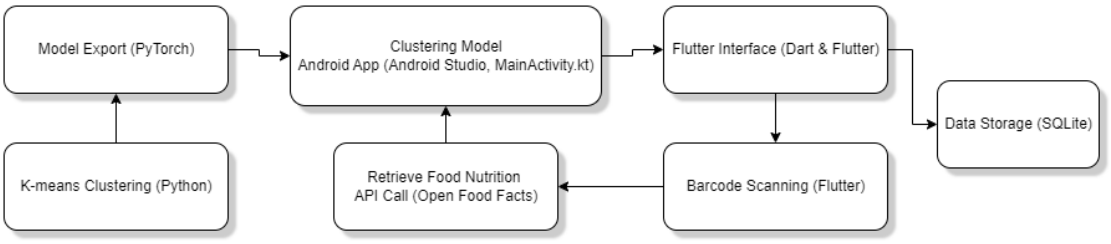


FIGURE 5.1: High Level Architecture Diagram

5.2.0.1 (i) Architecture of Your Solution

Original Design: The initial architecture was designed around utilising OCR technology to extract data directly from images of food labels.

Final Project: The architecture evolved to incorporate a more integrated mobile application that uses barcode scanning to fetch data from an API and applies a machine learning model for product classification.

Justification: The shift from OCR to barcode scanning was primarily due to the complexity and substantial time required to achieve high accuracy with OCR on diverse food label designs. Barcode scanning streamlined the data collection process and enhanced the reliability of the information retrieved.

5.2.0.2 (ii) List of Use Cases

Original Design: Use cases initially focused on capturing image data, processing it through OCR, and displaying nutritional information.

Final Project: Updated use cases now involve scanning barcodes, retrieving data from the Open Food Facts API, and classifying products based on nutritional data using the PyTorch clustering model.

Justification: The difficulties with OCR's accuracy and performance necessitated a shift to barcode scanning, which directly impacted the use cases to better suit the new method of data retrieval and analysis.

5.2.0.3 (iii) Risk Assessment

Original Design: Risks were primarily associated with the OCR's ability to handle varied label formats and potential inaccuracies in text extraction.

Final Project: The focus of risk assessment shifted to API reliability and the performance of the clustering model on mobile devices.

Justification: The change in technology from OCR to barcode scanning and API usage introduced new risks related to external data sources and computational demands of running machine learning models in a mobile environment.

5.2.0.4 (iv) Methodology to Develop Your Solution Approach

Original Design: The methodology was set to follow a traditional software development lifecycle, focusing on iterative improvements to the OCR technology.

Final Project: Adopted a more agile development approach, focusing on rapid prototyping and testing of barcode scanning and machine learning integration.

Justification: Encountering significant challenges with OCR and deciding to switch to a barcode-based system required a more flexible and iterative development approach to accommodate changes quickly and effectively.

5.2.0.5 (v) Implementation Schedule

Original Design: Scheduled to allow significant time for developing and refining OCR technology.

Final Project: The timeline was adjusted to accommodate the development of new components such as barcode scanning and integration with the machine learning model.

Justification: The pivot to a new solution approach necessitated a realignment of the project timeline to focus on developing and integrating new technologies.

5.2.0.6 (vi) Evaluation Plan

Original Design: Focused on evaluating the accuracy and efficiency of OCR in extracting nutritional data.

Final Project: Revised to evaluate the accuracy of barcode scanning, the reliability of data from the API, and the effectiveness of the clustering model.

Justification: The shift in technology required a new set of evaluation criteria to ensure the new components were meeting project goals effectively.

5.2.0.7 (vii) Prototype of the Resulting Product

Original Design: Envisioned a prototype demonstrating OCR capabilities.

Final Project: The final prototype features a mobile app with barcode scanning, API integration, and a user interface displaying classified nutritional information.

Justification: The difficulties in effectively implementing OCR led to the development of a more robust and user-friendly prototype using barcode scanning and machine learning, which better meets the project's objectives.

5.3 Transition in Project Scope and Detailed Solution Approach

Due to substantial changes in the project's direction from the initial design to the final implementation, a direct comparison between the original plan and the final project, as outlined in the "Actual Solution Approach", is not feasible. The project has evolved significantly during its development, leading to a redefined scope and objectives. The initial project design was significantly revised to better align with practical capabilities and emerging needs identified during the early stages of development. This shift was driven by both technological advancements and a strategic reassessment of the most effective ways to achieve the project's goals. As such, the final project bears little

resemblance to the original proposal, necessitating a fresh approach to documenting the implementation process. Instead of attempting to align the vastly different initial and final designs, this section will provide a detailed explanation of how the final project was developed. This includes a thorough description of each major phase of the project:

1. **Dataset Choice and Cleaning:** Beginning with the selection of the Open Food Facts dataset, this phase covers why this dataset was chosen and the extensive data cleaning processes undertaken to prepare the data for analysis.
2. **Clustering Process:** Details on how the data was clustered, including the methods used to determine the optimal number of clusters and the application of clustering algorithms.
3. **Model Conversion to PyTorch:** Explanation of how the clustering model was converted from a Scikit-Learn based implementation to PyTorch, preparing it for deployment in a mobile application environment.
4. **Mobile Application Development and Model Integration:** Final phase involving the development of the mobile application and the integration of the PyTorch clustering model into this application to provide real-time nutritional analysis.

5.4 Data Cleaning Process

5.4.1 Introduction to the Dataset

For my final year project, I selected the comprehensive dataset from Open Food Facts, which contains detailed information on over three million food products globally. This dataset is particularly valuable for nutritional analysis because of its extensive coverage of various product types and the depth of data provided on each item.

5.4.2 Rationale for Dataset Choice

The decision to use the Open Food Facts dataset was driven by its open-source accessibility and its broad adoption in nutritional studies, making it a reliable foundation for developing a food classification system based on nutritional content. The vast size and diversity of the dataset also offer a robust platform for applying complex data science techniques, such as machine learning clustering, to uncover dietary patterns.

5.4.3 Drawback of Using the Open Food Facts Database

A notable drawback of using an open-source database like Open Food Facts is that it allows public contributions, which can introduce errors or inconsistencies in the data, such as outliers or incorrectly entered values. These issues necessitate rigorous data cleaning to ensure the accuracy and reliability of the analysis.

5.4.4 Description of Data Cleaning

Data cleaning was a pivotal initial step to prepare the dataset for effective analysis. Given the size and open-source nature of the Open Food Facts dataset, this stage focused on enhancing data quality by:

- **Removal of Irrelevant Columns:** Numerous columns that were irrelevant to the core nutritional analysis, such as those pertaining to packaging and branding, were programmatically removed to narrow the dataset to nutritionally relevant variables.
- **Handling Missing Values:** The dataset's vast size meant that many records had incomplete data entries. Entries lacking critical nutritional information were excluded to ensure the machine learning model was developed from complete and accurate data.
- **Outlier Detection and Removal:** Outliers, which could skew analysis, were identified and removed using statistical methods. This was crucial to maintaining the integrity of the dataset's central tendencies and variabilities.

5.4.5 Conclusion

Automating the data cleaning process was essential given the dataset's extensive size (nearly 4GB). Manually cleaning such a vast array of entries would be impractical and prone to errors. Scripting the data cleaning ensures a standardised approach, reducing the risk of inconsistencies and enabling reproducibility. Effective data cleaning not only removes outliers and incomplete records but also significantly reduces the dataset size, making the clustering process more manageable and efficient. This streamlined dataset facilitates more accurate clustering by focusing on the most relevant and reliable data.

5.5 Clustering Processes

5.5.1 Principal Component Analysis (PCA)

Overview: Principal Component Analysis (PCA) is employed to reduce the dimensionality of large datasets whilst preserving as much variance as possible. It transforms the original variables into a new set of variables, which are linear combinations of the original variables and orthogonal to each other.

Description: In the project, PCA was applied to the nutritional data from the Open Food Facts dataset, which includes detailed metrics such as carbohydrates, fats, proteins, and sugars. This reduction in dimensionality was crucial to enhancing the performance and interpretability of the subsequent clustering algorithm applied.

Code Snippet Explanation:
This snippet demonstrates the scaling of features to ensure each variable contributes equally to the analysis, followed by the application of PCA to retain 95

```
# Prepare features for PCA, excluding the exceptions
features_to_use = df.drop(columns=exceptions, errors='ignore')

# Scale features using MinMaxScaler
scaler = MinMaxScaler()
features_scaled = scaler.fit_transform(features_to_use)

print("Features used for scaling and PCA:")
print(features_to_use.columns.tolist())

# Perform PCA with dynamic component selection
pca_full = PCA()
pca_full.fit(features_scaled)
cumulative_variance = np.cumsum(pca_full.explained_variance_ratio_)
n_components_needed = np.where(cumulative_variance >= variance_threshold)[0][0] + 1


# Fit PCA
pca = PCA(n_components=n_components_needed)
features_pca = pca.fit_transform(features_scaled)
```

FIGURE 5.2: Code snippet of scaling and PCA

5.5.2 Determination of Optimal Cluster Size

Overview: Choosing the right number of clusters for the K-Means clustering algorithm is essential to achieve meaningful segmentation of the dataset. An incorrect number of

clusters can lead to overfitting or underfitting, affecting the interpretability and usefulness of the results.

Description: To determine the optimal number of clusters for the nutritional data from the Open Food Facts dataset, I employed the elbow method, which involves plotting the sum of squared distances (inertia) for a range of cluster numbers and selecting the point where the rate of decrease sharply shifts (the elbow point).

Code Snippet Explanation:
This function loads the dataset, selects relevant nutritional features excluding non-feature columns like product names or codes, scales these features, and then calculates the inertia for different numbers of clusters. The inertia values are plotted against the number of clusters to visually determine the elbow point.

```
# Scale the features
scaler = MinMaxScaler()
features_scaled = scaler.fit_transform(features)

# Calculate KMeans for different values of k
sum_of_squared_distances = []
for k in range_k:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans = kmeans.fit(features_scaled)
    sum_of_squared_distances.append(kmeans.inertia_)

# Plotting the Elbow Method graph
plt.figure(figsize=(10, 6))

# Plotting sum of squared distances
plt.plot( *args: range_k, sum_of_squared_distances, 'bx-', markersize=8, linewidth=2, color='blue')
plt.xlabel('Number of Clusters (k)')
plt.ylabel('Sum of Squared Distances (Inertia)')
plt.title('Elbow Method For Optimal Number of Clusters (k)')
plt.grid(True)
plt.show()
```

FIGURE 5.3: Code snippet to determine optimal Cluster

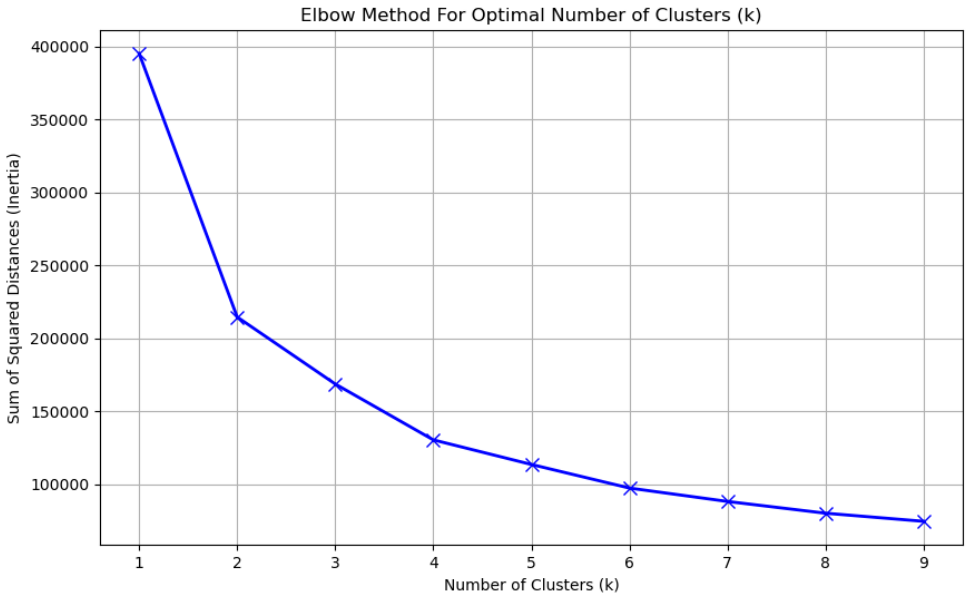


FIGURE 5.4: Graph of the Elbow Method

5.5.3 Clustering and Visualisation

Overview: After reducing dimensionality and determining the optimal number of clusters, the data is segmented into distinct groups using the KMeans algorithm.

Description: The clustering results are visualized to ensure that distinct, meaningful groups have been formed, facilitating easier interpretation and analysis. The visualisation plots the clusters based on the first two principal components.

Code Snippet Explanation:

```
# Perform K-means clustering
kmeans = KMeans(n_clusters=n_clusters, init='k-means++', random_state=42)
cluster_labels = kmeans.fit_predict(features_pca)
```

FIGURE 5.5: Code snippet of KMeans Clustering

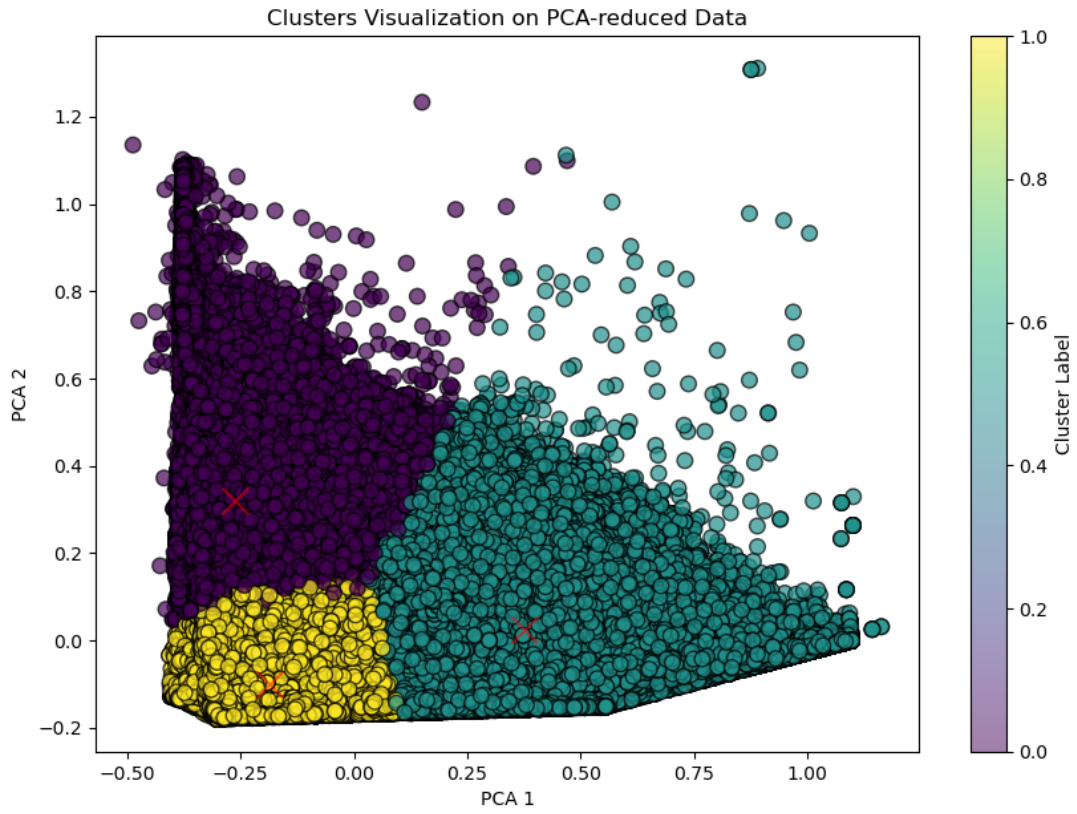


FIGURE 5.6: Scatter plot visualising Clusters using Principle Component 1 and Principle Component 2

This snippet performs the clustering and visualizes the clusters in a scatter plot, colour-coded by cluster label, providing a visual confirmation of the data segmentation.

Conclusion: These steps in the clustering process ensure that the dataset is optimally prepared and segmented for further analysis, such as nutritional evaluation or dietary recommendations. This methodical approach highlights the application of advanced data science techniques to real-world datasets in the nutritional domain.

5.6 Model Conversion to PyTorch

Overview:

The conversion of the clustering model to PyTorch involved creating custom PyTorch modules that replicate the functionality of the original Scikit-Learn model, but in a format compatible with mobile platforms. This step was crucial to facilitate deployment within a mobile application, leveraging PyTorch's computational efficiency and ease of integration.

Custom Modules Explanation:

1. Custom Scaler:
   The CustomScaler module scales input features using predefined minimum values and scale factors (derived from the MinMaxScaler in Scikit-Learn), ensuring that the data fed into the PCA has the correct scale.

```
class CustomScaler(nn.Module):
    Michal Strzelecki
    def __init__(self, scale, min_):
        super(CustomScaler, self).__init__()
        self.scale = torch.tensor(scale, dtype=torch.float32)
        self.min = torch.tensor(min_, dtype=torch.float32)

    Michal Strzelecki
    def forward(self, x):
        return (x - self.min) / self.scale
```

FIGURE 5.7: Code snippet of Custom Scaler for PyTorch Model

2. Custom PCA:
   The CustomPCA module reduces the dimensionality of the data by projecting it onto the principal component space defined by previously determined PCA components and mean.

```
class CustomPCA(nn.Module):
    Michal Strzelecki
    def __init__(self, components, mean):
        super(CustomPCA, self).__init__()
        self.components = torch.tensor(components, dtype=torch.float32)
        self.mean = torch.tensor(mean, dtype=torch.float32)

    Michal Strzelecki
    def forward(self, x):
        return torch.matmul(x - self.mean, self.components.T)
```

FIGURE 5.8: Code snippet of Custom PCA for PyTorch Model

3. Custom KMeans:
   The CustomKMeans module assigns data points to the nearest cluster center, a process central to the KMeans algorithm. This module calculates the Euclidean distance from each point to all cluster centers to determine the closest one.

```
class CustomKMeans(nn.Module):
    Michal Strzelecki
    def __init__(self, centers):
        super(CustomKMeans, self).__init__()
        self.centers = torch.tensor(centers, dtype=torch.float32)

    Michal Strzelecki
    def forward(self, x):
        expanded_centers = self.centers.unsqueeze(0).expand(x.size(0), -1, -1)
        expanded_x = x.unsqueeze(1).expand(-1, self.centers.size(0), -1)
        distances = torch.sqrt(torch.sum((expanded_x - expanded_centers) ** 2, 2))
        return torch.argmin(distances, dim=1)
```

FIGURE 5.9: Code snippet of Custom KMeans for PyTorch Model

Combining into a Pipeline:

These modules are integrated into a single PCAKMeansPipeline module, which processes input data through scaling, PCA transformation, and clustering in sequence.

```
class PCAKMeansPipeline(nn.Module):
    Michal Strzelecki
    def __init__(self, scaler, pca, kmeans):
        super(PCAKMeansPipeline, self).__init__()
        self.scaler = scaler
        self.pca = pca
        self.kmeans = kmeans

    Michal Strzelecki
    def forward(self, x):
        x_scaled = self.scaler(x)
        x_pca = self.pca(x_scaled)
        cluster_labels = self.kmeans(x_pca)
        return cluster_labels
```

FIGURE 5.10: Code snippet of Custom Pipeline Integration for PyTorch Model

Conclusion:

This comprehensive approach ensures that the model is not only accurate but also optimised for performance on mobile devices. The conversion process is capped off by

saving the entire pipeline model for deployment, demonstrating a successful translation of data processing and clustering logic from Scikit-Learn to PyTorch.

5.7 Mobile Application Development and Model Integration

5.7.1 Barcode Scanning

Overview:
Barcode scanning is a critical feature of the mobile application, enabling users to quickly and efficiently identify products by scanning their barcodes. This functionality is central to the app's operation, linking physical products to their digital data counterparts in the Open Food Facts database.

Implementation:
The application uses the BarcodeScannerService, a dedicated service within the app's architecture, to handle barcode scanning. This service integrates with the mobile device's camera hardware to capture barcode images and decode them to retrieve product identifiers.

Technical Details:
The barcode scanning process is initiated when the user opts to scan a product using the app's scanning interface. The service uses camera access permissions to activate the device's camera, captures the barcode image, and then employs decoding algorithms to translate the barcode into a machine-readable format.

Code Snippet Explanation:
This Dart code snippet demonstrates how the BarcodeScanningService uses the FlutterBarcodeScanner library to interact with the device's camera and perform real-time barcode scanning. The method scanBarcode is designed to be asynchronous, allowing the UI to remain responsive while waiting for the scanning process to complete.

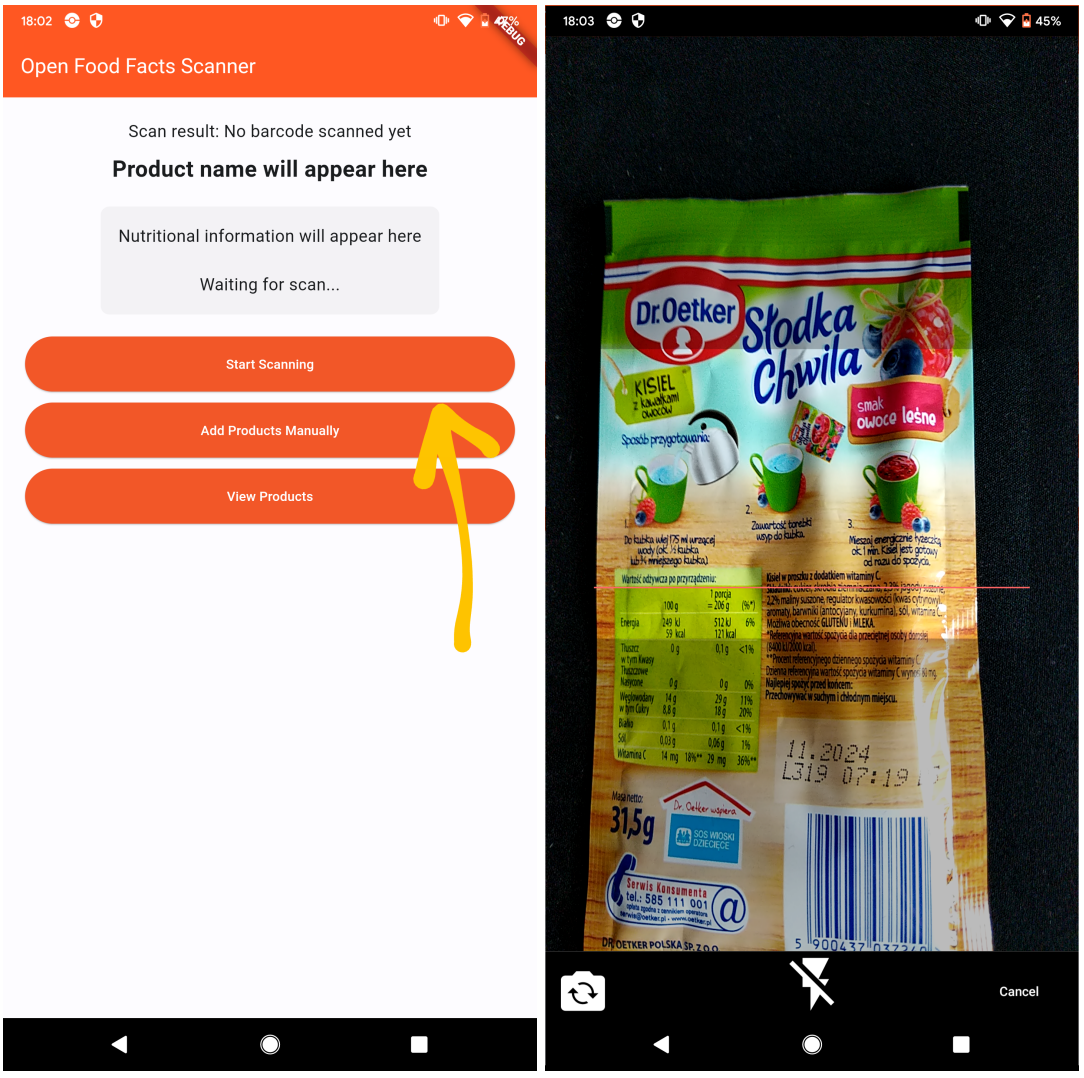
```dart
// barcode_scanner_service.dart
import 'package:flutter_barcode_scanner/flutter_barcode_scanner.dart';

class BarcodeScannerService {
  static Future<String?> scanBarcode() async {
    try {
      String barcodeScanRes = await FlutterBarcodeScanner.scanBarcode(
          "#ff6666", "Cancel", true, ScanMode.BARCODE);
      if (barcodeScanRes == '-1') {
        return null;
      }
      return barcodeScanRes;
    } catch (e) {
      print("Barcode scan error: $e");
      return null;
    }
  }
}
```

FIGURE 5.11: Code Snippet of Barcode Scanner Service

User Experience:

Upon initiating a scan, users are directed to a camera viewfinder screen, where they can align a barcode within a designated area. Once the barcode is detected and decoded, the application automatically retrieves the corresponding product data from the database or prompts the user if the scan was unsuccessful.



(a) Main Menu (b) Barcode Scanner View

FIGURE 5.12: Application Barcode Scanner

Integration with Product Data Retrieval:

Following a successful scan, the decoded barcode serves as a key to query the Open Food Facts API or the local product database, fetching detailed product information that includes nutritional values and other relevant data.

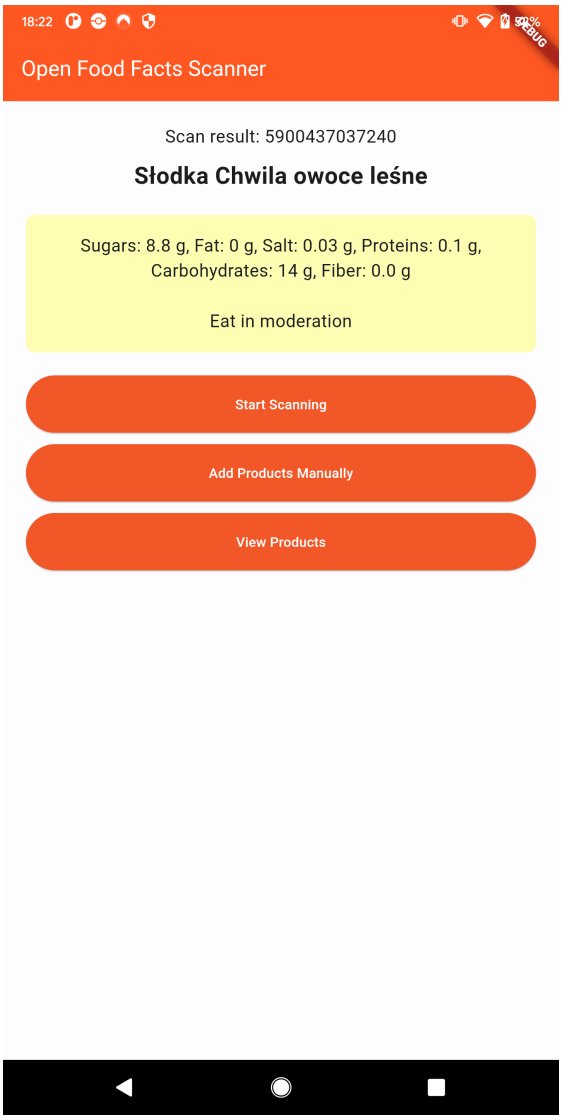


FIGURE 5.13: Barcode Scanner Results

Conclusion:

The barcode scanning functionality enhances the user experience by simplifying product identification but also serves as the first step in a chain of processes that provide valuable nutritional insights.

5.7.2 API Service Integration

Overview:

The API Service in the mobile application is responsible for retrieving detailed product information from the Open Food Facts API. This integration allows the app to access a vast database of product data, which is essential for providing users with accurate nutritional information.

Implementation:
The ApiService class encapsulates all interactions with the Open Food Facts API. It handles the construction of requests, management of responses, and parsing of data to ensure that the application can utilize the information effectively.

Technical Details:
The service uses HTTP requests to communicate with the Open Food Facts API, sending the barcode as a query parameter and receiving a JSON response containing comprehensive product details.

Code Snippet Explanation:
This Dart code snippet shows how the ApiService class fetches product information using the barcode. The method fetchProductInfo constructs a URL with the barcode, makes an HTTP GET request, and parses the JSON response into a map that the application can use.

```dart
// api_service.dart
import 'package:http/http.dart' as http;
import 'dart:convert';

class ApiService {
  Future<Map<String, dynamic>> fetchProductInfo(String barcode) async {
    // Specify fields to include product name, nutrition grades, and nutriments
    final String fields = 'product_name,nutrition_grades,nutriments';
    final uri = Uri.parse('https://world.openfoodfacts.org/api/v0/product/$barcode.json?fields=$fields');
    final response = await http.get(uri, headers: {
      'User-Agent': 'FoodScanner/1.0 (michael.strzelecki@mycit.ie)',
    });

    if (response.statusCode == 200) {
      return jsonDecode(response.body);
    } else {
      throw Exception('Failed to load product data');
    }
  }
}
```

FIGURE 5.14: API Service Integration

User Experience:
Upon scanning a barcode, the app calls this service to fetch detailed information about the product. This includes nutritional facts, ingredients, and other relevant data, which are then displayed to the user in a comprehensible format.

Error Handling:
The service includes error handling to manage scenarios where the API is unreachable or the product is not found. This ensures that the app remains robust and provides user feedback in case of network issues or data retrieval failures.

Integration with the Mobile Application:
The ApiService is tightly integrated with the barcode scanning functionality. Once a barcode is scanned and decoded, this service is invoked to retrieve the corresponding product data. The fetched data is then used to populate the product details view and feed into the nutritional clustering model.

Conclusion:
The API Service is a pivotal component of the mobile application, enabling access to a comprehensive database of product information. This functionality not only enriches the user experience by providing valuable data but also supports the app's core objective of promoting informed dietary choices based on reliable nutritional information.

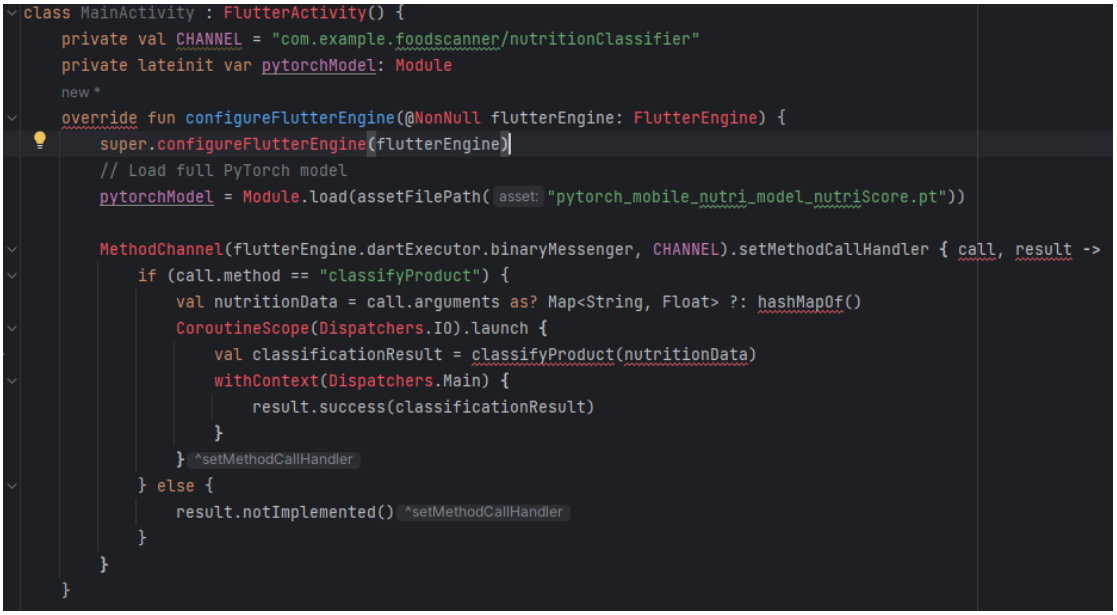
5.7.3 Integration of PyTorch Clustering Model

Overview:
The integration involves both the Dart front-end, which interacts with users, and the Kotlin back-end on Android, which handles more intensive computations including interaction with the PyTorch model.

5.7.3.1 Kotlin Side:

Model Loading and Method Channel Setup:
The PyTorch model is loaded in the Android environment using Kotlin. The MainActivity class sets up a MethodChannel that Flutter uses to communicate with native code. This channel listens for specific method calls from the Flutter side, particularly for classifying products based on their nutritional data.

Code Implementation:
This Kotlin code snippet demonstrates loading the PyTorch model and setting up a method channel that listens for classification requests. The classifyProduct function processes nutritional data received from Flutter, runs the PyTorch model, and returns the classification result.



```kotlin
class MainActivity : FlutterActivity() {
    private val CHANNEL = "com.example.foodscanner/nutritionClassifier"
    private lateinit var pytorchModel: Module
    new *
    override fun configureFlutterEngine(@NonNull flutterEngine: FlutterEngine) {
        super.configureFlutterEngine(flutterEngine)
        // Load full PyTorch model
        pytorchModel = Module.load(assetFilePath( asset: "pytorch_mobile_nutri_model_nutriScore.pt"))

        MethodChannel(flutterEngine.dartExecutor.binaryMessenger, CHANNEL).setMethodCallHandler { call, result ->
            if (call.method == "classifyProduct") {
                val nutritionData = call.arguments as? Map<String, Float> ?: hashMapOf()
                CoroutineScope(Dispatchers.IO).launch {
                    val classificationResult = classifyProduct(nutritionData)
                    withContext(Dispatchers.Main) {
                        result.success(classificationResult)
                    }
                } ^setMethodCallHandler
            } else {
                result.notImplemented() ^setMethodCallHandler
            }
        }
    }
```

FIGURE 5.15: PyTorch Model Integration Kotlin Side

5.7.3.2 Dart Side:

Flutter Integration:
On the Flutter side, the application interacts with this method channel to send nutritional data for classification and to receive the classification result, which influences the user interface.

```dart
// Function to classify food products
Future<void> classifyProduct(Map<String, dynamic> nutritionData) async {
  try {
    final int result = await platform.invokeMethod('classifyProduct', nutritionData);
    print("Product classified into cluster: $result");
    setState(() {
      _classificationResult = result;
      if (result == 0) {
        _classificationMessage = "Eat in moderation";
      } else if (result == 1) {
        _classificationMessage = "This is bad for you";
      } else if (result == 2) {
        _classificationMessage = "This is good for you";
      } else {
        _classificationMessage = "Classification error";
      }
    });
  } on PlatformException catch (e) {
    print("Failed to classify product: '${(e).message}'.");
  } catch (e) {
    print("Failed to classify product: 'Unknown error occurred.'");
  }
}
```

FIGURE 5.16: PyTorch Model Integration Dart Side

Conclusion:
The integration of the PyTorch clustering model into the mobile application is a crucial feature that significantly enhances its functionality. By leveraging Kotlin and Dart, the app effectively handles computational tasks on the native side and provides a responsive user interface on the front end. This setup allows the app to offer real-time, actionable nutritional advice based on sophisticated machine learning analysis, all within a mobile environment that is accessible to everyday users.

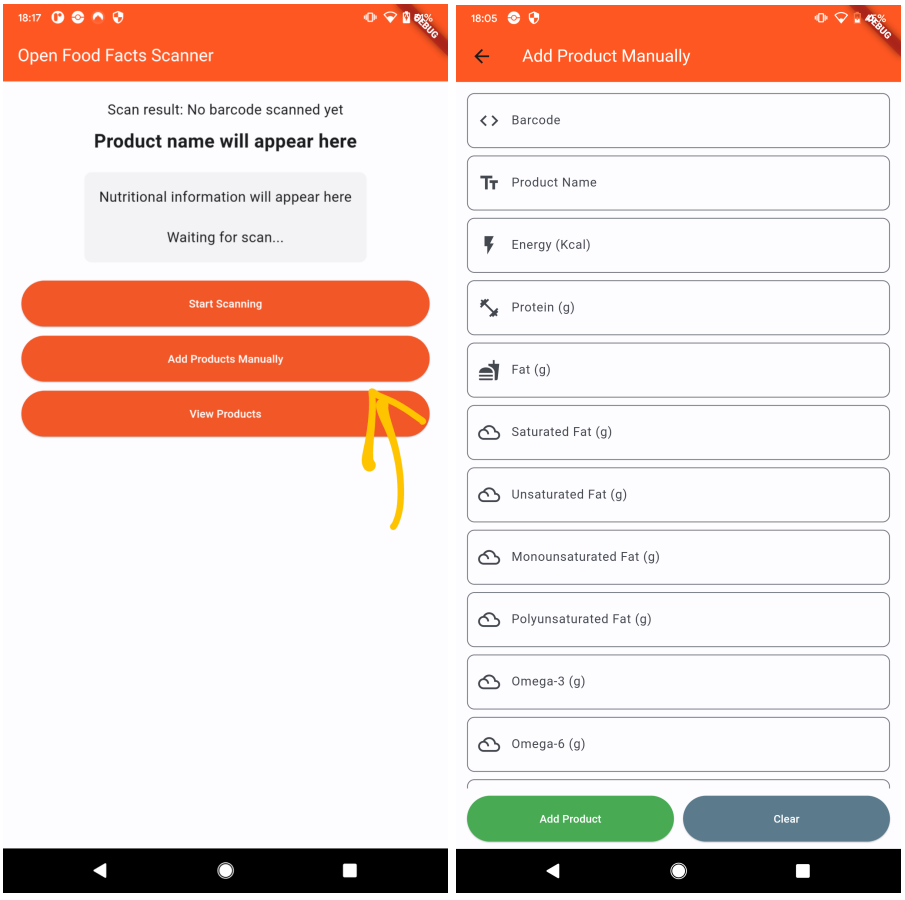
5.7.4 Manually Adding a Product

Overview:
The ability to manually add product information is an essential feature of the mobile application. It allows users to input detailed nutritional data about food products that

may not be available through barcode scanning. This is particularly useful for locally produced or unregistered products.

Functionality Overview:
The feature is implemented through a dedicated form within the app where users can enter various nutritional details about a product. The data collected includes macronutrients (such as fats, proteins, and carbohydrates), micronutrients (like vitamins and minerals), and other specific attributes (e.g., fiber, sugar types, etc.).

User Interaction:
Upon selecting the option to add a product manually, the user is presented with a comprehensive form. This form includes fields for all the relevant nutritional details that the app tracks. Each field is designed to be intuitive and guides the user on the type of information required.



(a) Main Menu (b) Barcode Scanner View

FIGURE 5.17: Application Manual Data Entry Screen

Process Description:
When a user fills out the form and submits the data, the app performs validation checks

to ensure all mandatory fields are correctly filled. The data is then saved to the local database, enabling the user to retrieve or analyze it later just like any other product scanned via barcode.

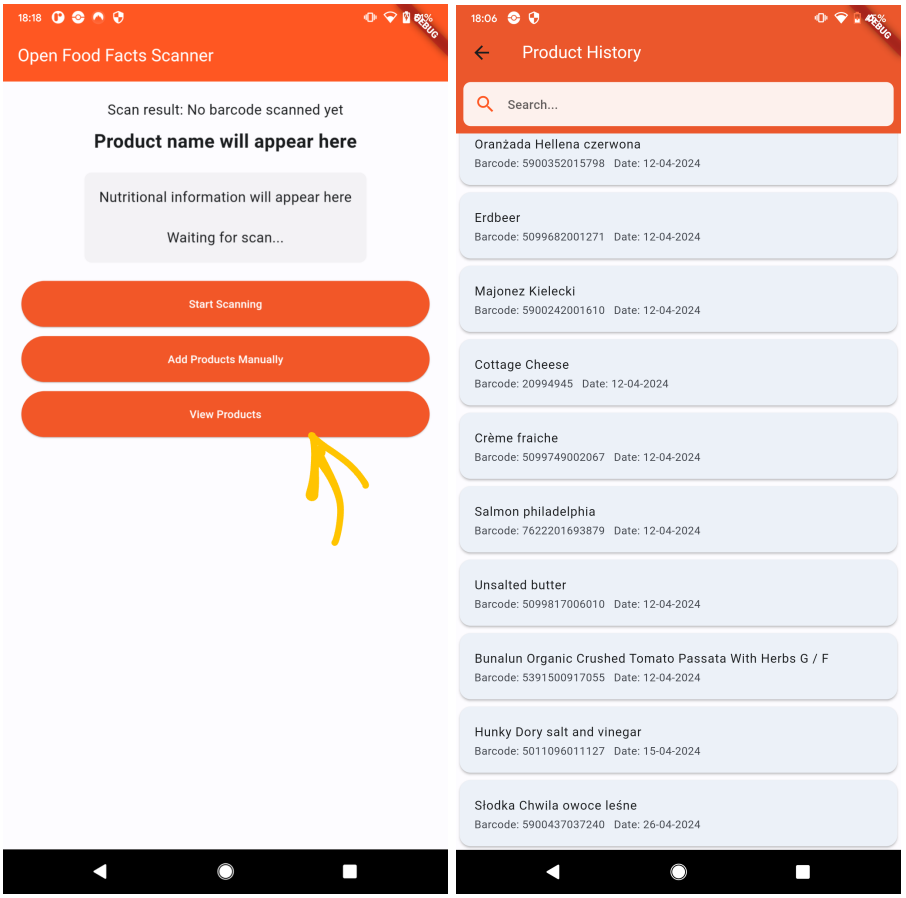
Integration with Clustering Model:
Once the product is added, the nutritional data can be immediately classified using the integrated PyTorch clustering model. This classification helps in providing instant feedback about the healthiness of the product based on its nutritional content.

Conclusion:
The capability to manually input product data significantly enriches the application's versatility and user engagement. This feature utilises the app's nutritional analysis tools, irrespective of whether the product is already catalogued in the global database.

5.7.5 Product History

Overview:
The Product History is a vital non-functional yet crucial aspect of the mobile application, providing a log of user interactions with various food products. While not central to the app's primary function of nutrition analysis, it plays a significant role in enhancing user experience by keeping a record of past entries.

Functionality and User Benefits:
The history feature acts as a repository of the user's scanned and manually entered products. It offers a chronological view, giving users the ability to revisit their product interactions. Notably, each entry in the history can be tapped to reveal more detailed nutritional information. Moreover, users have the option to re-evaluate the product using the integrated clustering model, ensuring they have the most current classification based on their health and dietary preferences.



(a) Main Menu (b) History View

FIGURE 5.18: Application History Screen

User Experience Enhancement:

The addition of a search bar at the top of the Product History screen significantly streamlines user interaction by allowing for quick product retrieval. Users can effortlessly locate a specific item by entering relevant keywords, which is particularly beneficial when the history log grows over time.

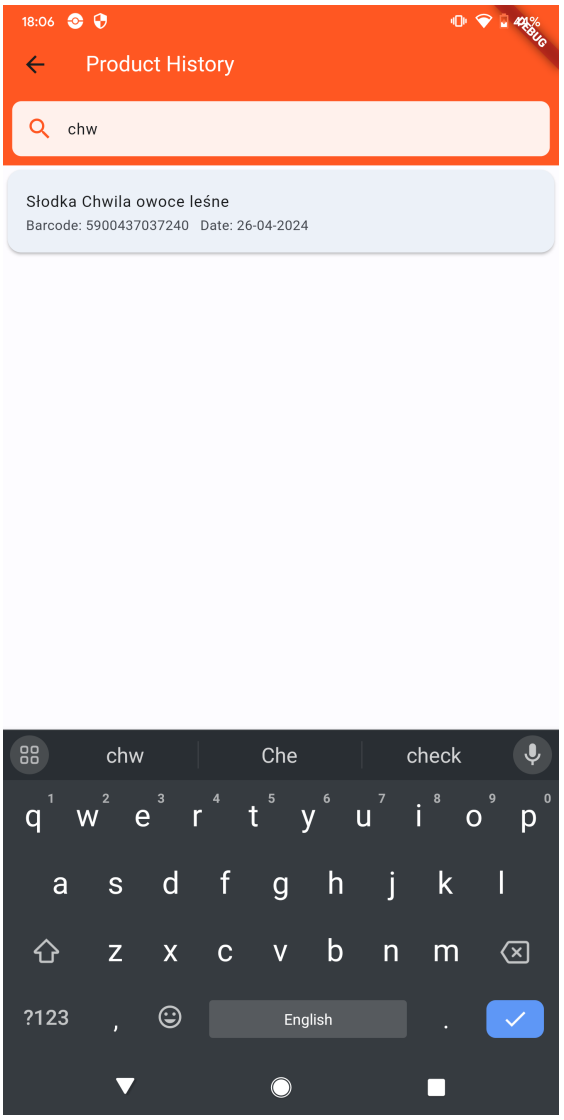
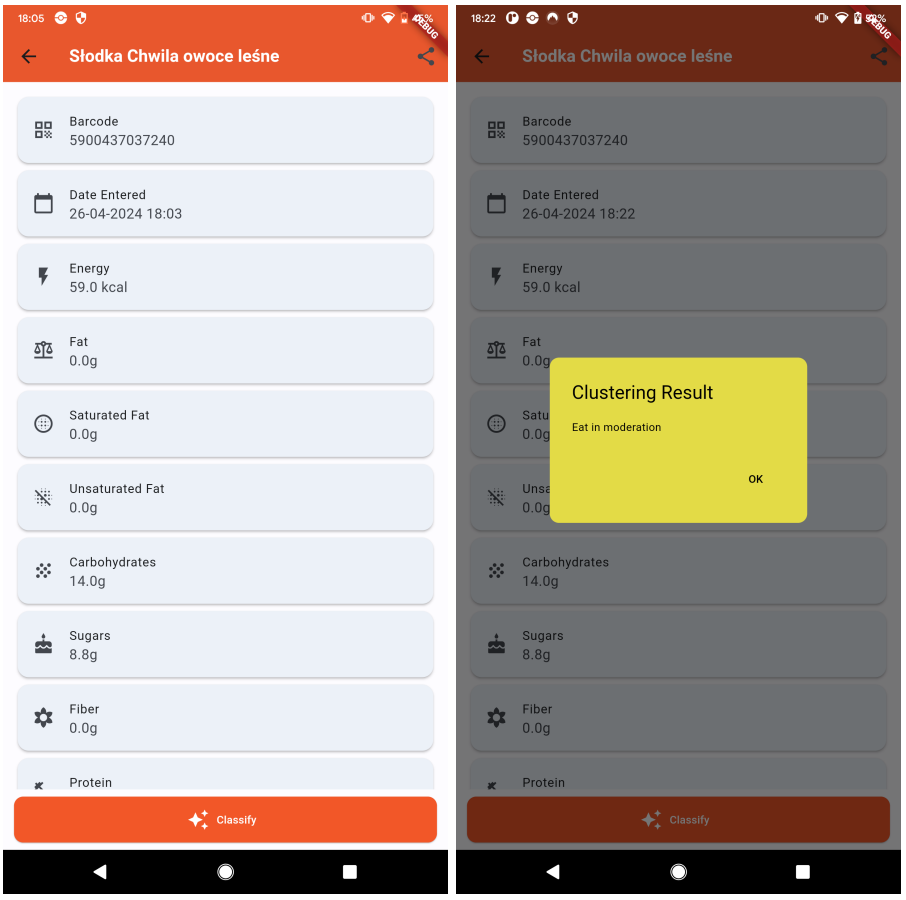


FIGURE 5.19: Application History Screen Showcasing Search Function

Integration and Interactivity:

Upon selecting a product from the history list, users are presented with an in-depth view of the product's details. This view is not just a static display but an interactive interface that invites users to re-run the product classification. Such functionality reaffirms the app's dynamic nature, promoting continuous user engagement.



(a) Detailed History View (b) Product Classification View

FIGURE 5.20: Application Detailed History Screen and Product Classification

Technical Implementation Insights:

The technical architecture of the history feature includes interfacing with a SQLite database to store and manage product records and employing Flutter widgets for the user interface. A significant aspect of this architecture is the integration of the database operations with the app's front end, ensuring that data retrieval and manipulation are both efficient and reliable.

Conclusion:

While the Product History may not be a direct functional component of the nutritional analysis process, its presence is a testament to the thoughtfulness of the application's design. It enriches the user experience by providing a personalised and detailed dietary log, facilitating easy access and management of historical data. The ability to reclassify products underscores the dynamic and user-centered approach of the application, highlighting the project's commitment to delivering a comprehensive and valuable tool for personal health management.

Chapter 6

Testing and Evaluation

6.1 Metrics for Model Evaluation

6.1.1 Introduction:

In the realm of clustering analysis, the quality and validity of the generated clusters are paramount. The metrics chosen to evaluate the clustering model were determined by the specific requirements of the project and the nature of the nutritional data under study.

6.1.2 Metrics Selection:

6.1.2.1 Explained Variance Ratio:

The primary quantitative metric utilised was the explained variance ratio from the Principal Component Analysis (PCA). This ratio was pivotal to determine the number of components to retain for a robust clustering outcome, ensuring that the reduced dimensions still encapsulate the essence of the original data with minimal loss of information.

6.1.2.2 Visualisation of Clusters:

Visually assessing the separation and cohesion of clusters is an intuitive metric. By plotting the clusters in a two-dimensional space, we could observe the grouping efficacy post-PCA reduction and KMeans clustering. Although this visual evaluation is not strictly quantitative, it provided an immediate, perceptible understanding of cluster quality.

6.1.2.3 Inertia Graphing:

While not inspecting the inertia directly, the algorithm's inertia was charted to help identify the 'elbow' in the graph, indicating an optimal number of clusters. This graphical method strikes a balance between cluster compactness and the number of clusters, contributing to the decision-making process for selecting an appropriate number of clusters value.

6.1.2.4 Manual Inspection:

Complementing the visual methods, a manual review of the clustering results was performed. This involved inspecting the cohesiveness of data within clusters and ensuring clear demarcation between different clusters, thus establishing qualitative assurance of the model's practical output.

6.1.3 Word Cloud Visualisation:

To qualitatively evaluate the characteristics of each cluster, word clouds were employed. By generating word clouds, it was possible to visually discern the most prevalent features within each cluster, providing an intuitive understanding of the cluster's composition. This visualisation technique allowed for a straightforward interpretation of the clustering model's output, highlighting the dominant nutritional attributes that characterise each group.

6.1.4 Summary Statistics and Graphical Analysis:

Summary statistics also played a role in assessing the clusters. For each cluster, key nutritional features were aggregated, and their mean values were computed. These mean values were then graphically represented to compare the clusters side by side. This approach provided a clear, at a glance understanding of how each cluster differed in terms of nutritional content, offering an empirical basis for the model's effectiveness in grouping similar products together.

6.1.4.1 Future Work:

The project's transition from its initial concept resulted in different goals and methodologies. Due to time constraints the use of additional metrics like the Silhouette Score,

which would have provided further insights into the distance between and within clusters. The possibility of delving deeper into inertia as a standalone metric was noted for subsequent development. Future iterations of the project would benefit from an expansive preparatory phase, determining a more comprehensive suite of evaluative measures and setting a baseline for expected outcomes

6.1.4.2 Conclusion:

Given the dynamic nature of the project's trajectory, the metrics were chosen and applied with a focus on pragmatic evaluation, rooted in the project's revised scope. The employed metrics while limited in number were aptly suited to deliver insightful assessments of the clustering model, guiding the project towards its ultimate objective: the effective categorisation of food products based on nutritional content.

6.2 System Testing of Clustering Model

Testing for the clustering model involved using a Python environment with libraries such as Pandas for data manipulation, Scikit-learn for implementing PCA and KMeans algorithms, and Matplotlib along with Seaborn for data visualisation. The dataset used was a comprehensive compilation of nutritional information obtained from the Open Food Facts database.

6.2.1 Metrics and Measurements

The performance of the clustering model was evaluated using specific metrics, each chosen for its relevance to the project's goals:

6.2.1.1 PCA Explained Variance:

Purpose: Measures the proportion of the dataset's variance that each principal component retains, crucial for understanding the impact of reducing dimensionality on data integrity.
Measurement Method: After applying PCA, the explained variance ratio for each principal component was calculated and cumulatively summed to determine how many components were needed to retain at least 95% of the total variance.
Inputs: The normalized dataset with selected nutritional features, prepared using Pandas and pre-processed to ensure data quality.

6.2.1.2 Heatmap of PCA Loadings:

Purpose: To visually represent the contribution of each feature to the principal components, highlighting how different variables influence the construction of components.
Measurement Method: A heatmap was generated using Seaborn, which visually displayed the loadings of each feature on the principal components. This visualization helps in understanding which features most strongly affect the dimensions and are thus most important in the clustering process.
Inputs: The PCA results including component loadings were used to create the heatmap.

6.2.1.3 Elbow Method for Optimal Clusters:

Purpose: Aids in selecting the optimal number of clusters by identifying the point where the addition of another cluster does not give significantly better modeling of the data.
Measurement Method: KMeans clustering was performed for a range of cluster numbers, and the inertia (sum of squared distances from each point to its assigned center) for each was plotted. The 'elbow'—the point where the inertia's rate of decrease sharply changes—indicated the optimal number of clusters.
Inputs: The PCA-reduced data from Scikit-learn's PCA transformation was used for clustering.

6.2.1.4 Cluster Visualization:

Purpose: Provides a visual assessment of the clustering quality, showing how well the data points are grouped within clusters and separated between clusters.
Measurement Method: Using the top two principal components, scatter plots were created to visualize the clusters. Each cluster was marked with a distinct color, and centroids were plotted to highlight the cluster organization.
Inputs: The PCA-reduced dataset and the clustering labels generated by the KMeans algorithm.

6.2.2 Inputs for Experiments

Inputs were meticulously prepared to ensure they accurately represented the analytical needs of the project:

6.2.2.1 Dataset Preparation:

Data Cleaning: Involved removing entries with excessive zeros and outliers that could skew the results. Nutritional features were normalized using MinMaxScaler to ensure no single feature would dominate due to scale differences.
Feature Selection: Critical nutritional features were selected based on their relevance to health and dietary guidelines, ensuring the model's focus was aligned with project objectives.

6.2.2.2 PCA and Clustering Parameters:

PCA Components: The number of components retained was based on the explained variance, ensuring sufficient data complexity was preserved while reducing dimensionality.
Number of Clusters: Determined by the elbow method, this parameter was essential for ensuring the clusters were neither too granular nor too generalized.

6.2.3 Execution of Tests

Each step in the testing process was conducted methodically:

6.2.3.1 PCA Implementation:

The PCA was applied to the normalized data to reduce its dimensionality while retaining essential information. The number of components was dynamically chosen based on the explained variance.

6.2.3.2 Heatmap Generation:

Provided visual insight into which features most significantly impact each principal component, enhancing understanding of the PCA process.

6.2.3.3 KMeans Clustering Process:

The clustering was executed using the PCA-reduced data. The process was iterative, adjusting the number of clusters based on the elbow method's output to optimize the model's granularity and effectiveness.

6.2.3.4 Visualizations and Statistical Analysis:

Detailed visualizations were created to inspect the clustering results visually. Additionally, statistical measures like mean and variance were calculated for each cluster to analyze the nutritional profile comprehensively and validate the clustering logic.

6.3 Results and Analysis

6.3.1 PCA-Reduced Cluster:

The images 6.1 and 6.2 display clusters after PCA reduction. The clustering depicted here shows distinct groupings which reflect the variance in the data based on nutritional content. The visuals showcase clear separation among the clusters, which indicates a good clustering process with minimal overlap. The red 'x' marks represent cluster centroids, key in understanding the 'average' position of each cluster in the reduced-dimensional space. In terms of practicality, these visuals align well with the model's objective to differentiate food items nutritionally.

FIGURE 6.1: Clustering of PC1(Carbohydrates and Sugars) against PC2(Fat and Saturated Fat)

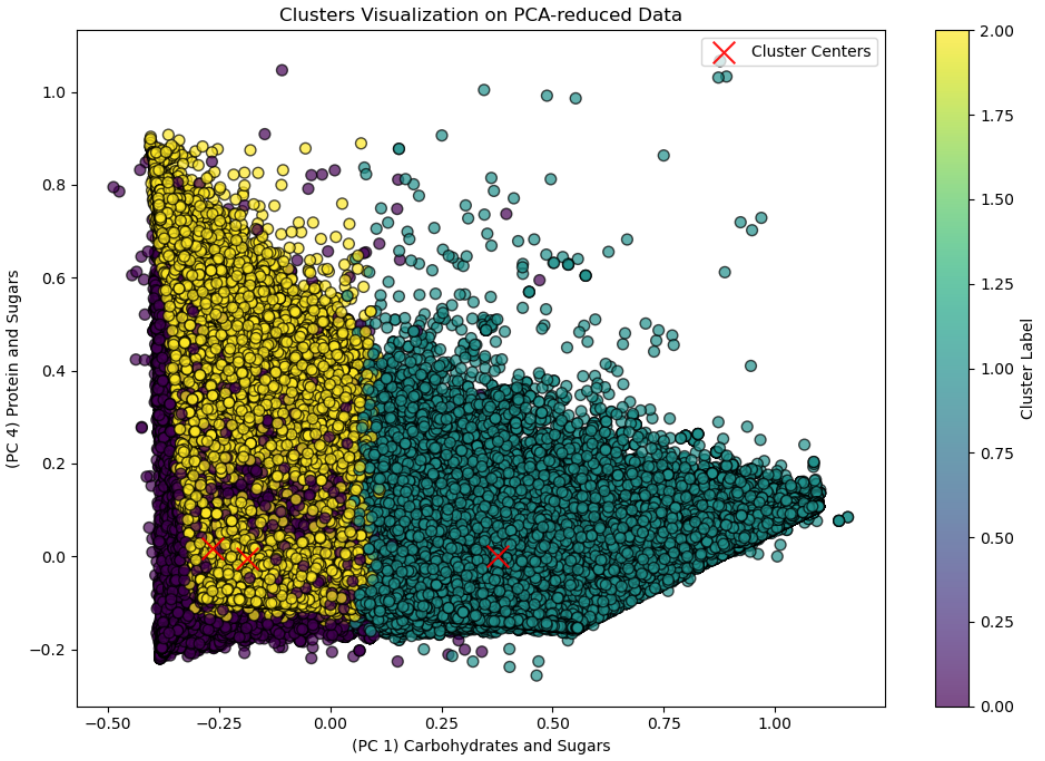


FIGURE 6.2: Clustering of PC1(Carbohydrates and Sugars) against PC4(Protein and Sugars)

6.3.2 Mean Nutritional Values per Cluster:

The 6.3 graph summarises the mean nutritional values for each cluster. This visualization is crucial for understanding the nutritional emphasis of each cluster, such as one being high in fats or another in proteins. Compared to other systems, this provides a direct nutritional profile comparison, which could be used in diet planning applications.

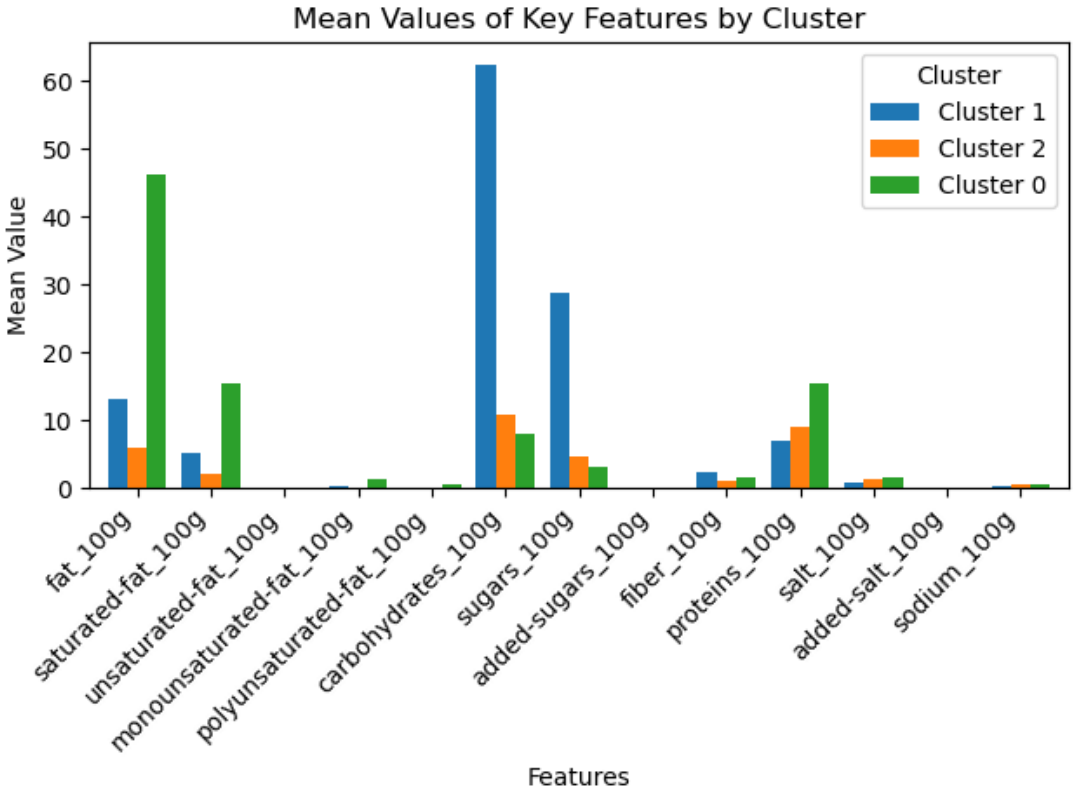


FIGURE 6.3: Mean Value of Key Features by Cluster

6.3.3 Word Clouds for Cluster Description:

Word clouds 6.4, 6.5, and 6.6 provide a qualitative insight into the composition of the clusters, displaying the most frequent terms associated with food items within each group. Larger words indicate higher frequency, which can be interpreted as the defining features or ingredients of that cluster. For example, the predominance of 'chocolate', 'butter', and 'cheese' in one cluster might suggest it groups together high-fat foods. This kind of analysis helps to quickly understand the dietary character of each cluster, and while not quantitative, it complements the statistical data well..

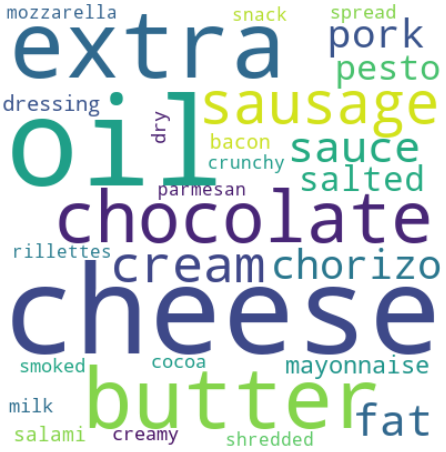


FIGURE 6.4: Cluster 1 Word-cloud

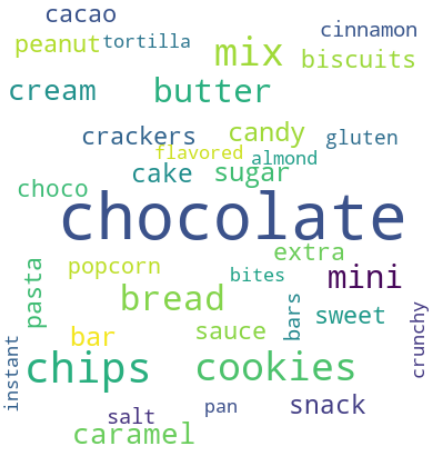


FIGURE 6.5: Cluster 1 Word-cloud

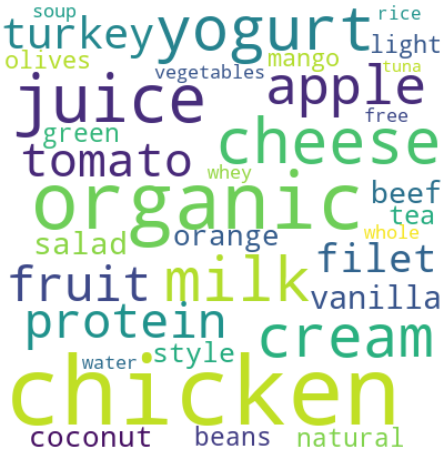


FIGURE 6.6: Cluster 1 Word-cloud

6.3.4 Feature Contribution Heatmap:

This heatmap 6.7 indicates how each feature contributes to the principal components derived during PCA. Red tones show positive contributions, while blue tones indicate negative contributions. This detailed view can identify which nutrients most influence the differentiation of food products in the reduced space. For instance, high positive values for sugars in a component suggest that sugars are a defining feature of the cluster aligned with that component. This graph provides an in-depth understanding of the feature relationships within the data.

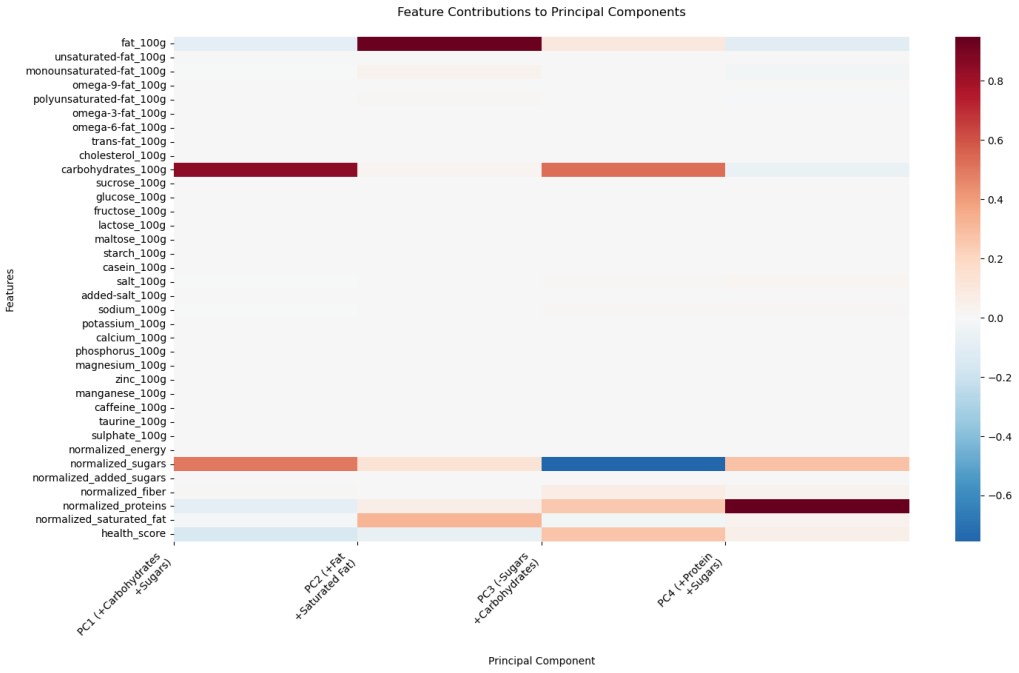


FIGURE 6.7: Feature Contributions to Principle Components

6.3.5 Explained Variance by PCA Components:

The bar graph 6.8 shows the individual and cumulative percentage of variance explained by each PCA component. This is crucial for justifying the number of dimensions retained post-PCA. A high cumulative variance explained by the first few components confirms that they retain most of the information from the original dataset. It ensures that the model remains comprehensive while becoming computationally efficient for deployment in a mobile application.

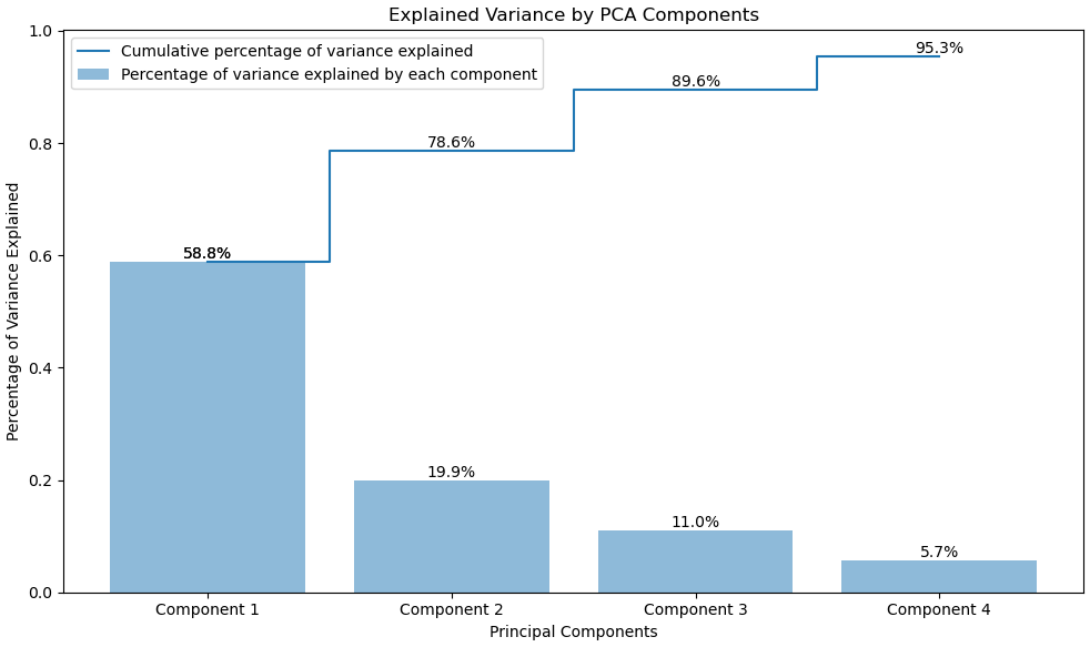


FIGURE 6.8: Explained Variance by PCA Components

6.3.6 Elbow Method for Cluster Determination:

The elbow plot 6.9 visualises the model's inertia as the number of clusters increases, helping to identify the optimal cluster count. The 'elbow' point—where the rate of decrease sharply changes—suggests the point beyond which adding more clusters doesn't give much better modelling of the data. The method is an established heuristic used in k-means clustering to guide the selection of an appropriate number of clusters.

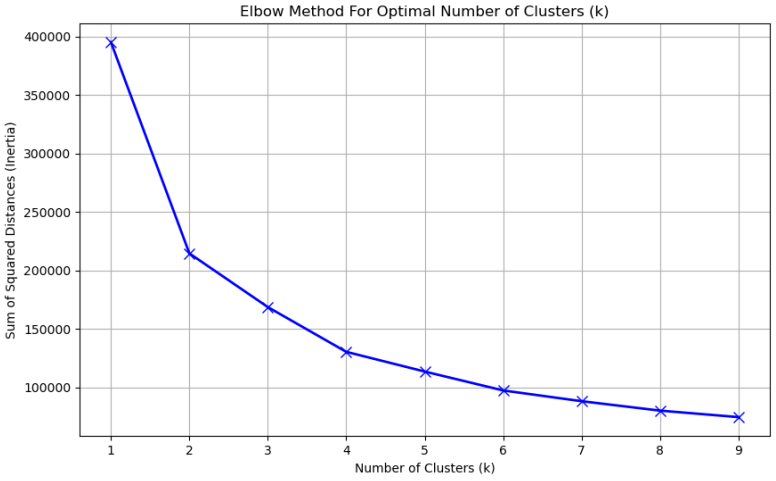


FIGURE 6.9: Elbow Method for Cluster Determination

6.3.7 Conclusion:

In addressing potential concerns surrounding this project, two primary threats to the integrity of our results merit discussion. Firstly, the interpretive nature of the word clouds could introduce a subjective bias, where the prominence and interpretation of terms may not always be quantitatively solid. Secondly, while the PCA method has successfully distilled our data into principal components, there is an inherent risk of oversimplification. Key variances in nutritional information may have been inadvertently discarded in the process, potentially obscuring nuanced differences between products.

Further to this, our reliance on the Open Food Facts database brings with it a set of challenges. The open-source framework, though rich in data, is susceptible to inaccuracies that are incumbent upon user-submitted information. The veracity of the data, therefore, cannot be unequivocally guaranteed.

To conclude, the set of statistical and graphical techniques applied in our evaluation has been instrumental in validating the efficacy of our clustering model. The model adeptly categorises food items based on their nutritional profiles, as substantiated through extensive visual analysis. Nevertheless, it's imperative to acknowledge that the objectivity of our results could be potentially compromised by the subjective interpretation of some analytical tools and the foundational data's accuracy.

Chapter 7

Discussion and Conclusions

7.1 Solution Review

The evolution of my final year project from its inception to completion has been a profound journey of adaptation and innovation. It began with the ambition to optimise OCR technology for reading nutritional labels, a task that quickly revealed itself to be a deep well of technical complexity due to the diverse and intricate designs of food packaging. The challenges of achieving high accuracy with OCR technology, while notable, led to a pivotal shift in the project's direction.

The project's trajectory was redirected towards developing a clustering model integrated within a mobile application, which ultimately provided a more practicable and efficient solution. The transition from OCR to the application of PCA and K-Means clustering for categorising nutritional data proved to be a significant leap towards usability and functionality.

The current iteration of the project, embodied in the mobile app, is a departure from the OCR-focused approach, aligning more closely with the end goal of making nutritional information readily available and understandable to users. The app simplifies the complex data into comprehensible groups, demonstrating the project's ability to evolve and address the problem statement in an innovative manner.

In the realm of nutrition, where data is abundant but often convoluted, the application serves as a bridge between dense information and the end-user, enabling healthier lifestyle choices. While the original objective was to refine OCR technology for a niche

application, the project's final form serves a broader purpose and reaches a wider audience, an outcome that underscores the dynamic nature of problem-solving in computer science.

In reflection, the shift from a technical challenge in OCR to a solution that prioritises user engagement and practicality underscores the project's responsiveness to the need for direct, accessible health-related insights. It exemplifies the essence of adaptability within the field of computer science and has culminated in a solution that not only solves the problem at hand but also paves the way for future explorations in the domain of digital health innovations.

7.2 Project Review

Looking back over the course of this project, it's evident that it has been an expansive learning curve that tested both my adaptability and technical skills. Beginning with the original ambition to refine OCR technology, the project's pivot to developing a mobile application with integrated clustering for nutritional data was both challenging and rewarding.

This shift in direction taught me the importance of flexibility in project management and the need to re-evaluate goals in response to technical and practical feasibility. The transition was a testament to the project's agile development process, where feedback and iterative testing were crucial in steering the project towards a more viable and impactful outcome.

The lessons learned were manifold. On a technical level, I gained proficiency in Python, particularly in the use of libraries such as sklearn for machine learning and matplotlib for data visualisation. Furthermore, the application of PCA and K-Means clustering algorithms enhanced my understanding of data science and its practical applications.

One of the most significant skills honed during this project was problem-solving. Each obstacle encountered, whether it was related to data cleaning or the integration of the clustering model into the mobile app, required a strategic and methodical approach. Learning to navigate these issues was instrumental in my professional development.

Additionally, the project has been a platform for honing my software development skills, especially in Flutter and Dart for app development, which was a new venture for me. The integration of PyTorch models into a mobile environment was another area that stretched my capabilities and will be invaluable for future projects.

If given the chance to start over, I would allocate more time to the initial planning phase to anticipate potential shifts in the project's direction. I would also incorporate user feedback earlier in the development cycle to ensure the app's features aligned closely with user needs.

The real-world problem-solving experience gained from this project has been immense. It has equipped me with a set of transferable skills that I am eager to apply in future endeavours. The project has also highlighted the importance of a user-centred approach in technology development, an insight I will carry forward into my career.

7.3 Conclusion

This project began with the aim to enhance the accessibility and understanding of nutritional information through the optimization of OCR technology. However, as the project unfolded, it evolved into the development of a mobile application that categorizes foods based on their nutritional content using a clustering model. This transition was not just a shift in technology but also a strategic pivot towards a more scalable and widely applicable solution.

- The clustering model successfully grouped food items into meaningful categories based on their nutritional profiles. This was achieved through the application of PCA for dimensionality reduction followed by K-Means clustering. The clusters were distinct and well-defined, which supports the model's effectiveness in segmenting foods in a way that can inform healthier eating choices.
- The mobile application proved to be a user-friendly platform that allows users to easily scan and get insights into their food choices. This tool not only makes the data more accessible but also more actionable, providing users with immediate feedback about the nutritional quality of their food.
- The project's evolution from an OCR-focused approach to a data clustering application highlighted the flexibility needed in software development. It demonstrated

the ability to adapt to new information and changing circumstances, which was crucial in redirecting the project towards a more viable outcome.

- The project allowed for the development and application of technical skills in real-world scenarios. Skills in data analysis, machine learning, and mobile application development were not only enhanced but also effectively applied to produce a functional and impactful tool.

In conclusion, this project has not only met its revised objectives but has also provided a foundation for further exploration and development in the field of digital health solutions. The ability to pivot and adapt to the project's needs has been a valuable learning experience, emphasizing the dynamic nature of technology and the importance of maintaining flexibility in problem-solving approaches.

7.4 Future Work

As the project transitions from its current state to future iterations, several enhancements and expansions are planned to broaden its scope and increase its efficacy:

1. Incorporating a Broader Range of Nutritional Data: Adding more detailed nutritional factors would enrich the analysis, allowing the app to cater to a variety of dietary needs and preferences. This could include tracking micronutrients like vitamins and minerals, which would appeal to users interested in comprehensive dietary planning.

2. Enhanced Personalization: Integrating machine learning algorithms to offer personalized dietary recommendations based on individual user profiles or specific health goals could significantly enhance user engagement and satisfaction. This personalization would make the app not just a tool for information but a daily companion for health management.

3. Increased Interactivity: Features that allow users to log their dietary habits, set nutritional goals, and receive feedback could transform the app into an interactive platform that actively supports users in achieving healthier lifestyles.

4. Refining Clustering Techniques: Future work could also explore the use of more advanced clustering algorithms and techniques to enhance the precision and accuracy of the food categorization. Implementing methods like hierarchical clustering or density-based clustering algorithms like DBSCAN could uncover more nuanced relationships within the data.

5. Dynamic Clustering Adaptation: Developing a dynamic model that adapts to new data or user feedback could make the clustering process more robust and responsive. This adaptive approach would allow the model to refine its classifications based on real-world usage and feedback, ensuring the clusters remain relevant and useful as dietary trends evolve.

6. Exploring Alternative Dimensionality Reduction Techniques: While PCA is effective, alternative techniques like t-SNE or UMAP could provide better visualization and interpretation of high-dimensional data, potentially revealing more insightful clustering patterns.

Bibliography

[1] TheAILearner, "Optical character recognition pipeline," https://theailearner.com/2019/05/28/optical-character-recognition-pipeline/, May 2019.

[2] D. Berchmans and S. S. Kumar, "Optical character recognition: An overview and an insight," in *2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT)*, 2014, pp. 1361–1365.

[3] I. Ozhmegov. (2023, Apr) Ocr technologies compared: Tesseract, abbyy ocr, google cloud vision, paddle ocr, easyocr, and keras ocr. [Online]. Available: https://medium.com/@ilia.ozhmegov/ocr-technologies-compared-tesseract-abbyy-ocr-google-cloud-vision-paddle-ocr-easyocr-and-6312c

[4] R. Mittal and A. Garg, "Text extraction using ocr: A systematic review," pp. 357–362, July 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9183326

[5] T. D. Piyadasa. (2022, June) Image binarization in a nutshell. [Online]. Available: https://medium.com/@tharindad7/image-binarization-in-a-nutshell-b40b63c0228e

[6] A. Boukharouba, "A new algorithm for skew correction and baseline detection based on the randomized hough transform," *Journal of King Saud University - Computer and Information Sciences*, vol. 29, no. 1, pp. 29–38, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1319157816300015

[7] R. Subudhi, B. Sahu, and P. Mohapatra, "A novel noise reduction method for ocr system," 12 2013. [Online]. Available: https://ijcst.com/vol5/spl2/ec1145.pdf

[8] Z. Raisi, M. A. Naiel, P. W. Fieguth, S. Wardell, and J. S. Zelek, "Text detection and recognition in the wild: A review," *ArXiv*, vol. abs/2006.04305, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:219531885

[9] V. M. U. de Strasbourg). https://vincmazet.github.io/bip/detection/edges.html. Accessed: 2023-10-18.

[10] A. Humeau-Heurtier, "Texture feature extraction methods: A survey," *IEEE Access*, vol. 7, pp. 8975–9000, 2019, hal-02126655.

[11] I. H. Sarker, "Machine learning: Algorithms, real-world applications and research directions," *SN Computer Science*, vol. 2, no. 3, 2021.

[12] I. Sarker, "Deep learning: A comprehensive overview on techniques, taxonomy, applications and research directions," *SN Computer Science*, vol. 2, p. 420, 2021.

[13] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang, "East: An efficient and accurate scene text detector," *arXiv preprint arXiv:1704.03155*, 2017.

[14] L. Alzubaidi, J. Zhang, A. Humaidi *et al.*, "Review of deep learning: concepts, cnn architectures, challenges, applications, future directions," *Journal of Big Data*, vol. 8, no. 53, 2021.

[15] R. Smith, "An overview of the tesseract ocr engine," in *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, vol. 2, Sep. 2007, pp. 629–633.

[16] G. Bradski *et al.*, "Opencv (open source computer vision library)," 2000. [Online]. Available: https://opencv.org/about/

[17] H. Software, "What is optical character recognition (ocr) technology?" *Hyland Resources*, 2023. [Online]. Available: https://www.hyland.com/en/resources/terminology/data-capture/what-is-optical-character-recognition-ocr

[18] M. Nadira, N. I. N. Kamariah, M. Z. Jasni, and A. B. S. Azami, "Optical character recognition by using template matching (alphabet)," 2007. [Online]. Available: https://api.semanticscholar.org/CorpusID:21037699

[19] M. Z. Hossain, M. A. Amin, and H. Yan, "Rapid feature extraction for optical character recognition," *ArXiv*, vol. abs/1206.0238, 2012. [Online]. Available: https://api.semanticscholar.org/CorpusID:14914708

[20] M. M. Taye, "Understanding of machine learning with deep learning: Architectures, workflow, applications and future directions," *Computers*, vol. 12, no. 5, 2023. [Online]. Available: https://www.mdpi.com/2073-431X/12/5/91

[21] L. Y. Data. (2023) Ocr with deep learning: How do you do it? [Online]. Available: https://labelyourdata.com/articles/ocr-with-deep-learning

[22] Z. Lei, S. Zhao, H. Song *et al.*, "Scene text recognition using residual convolutional recurrent neural network," *Machine Vision and Applications*, vol. 29, pp. 861–871, 2018.

[23] Z. Xiao, Z. Nie, C. Song, and A. T. Chronopoulos, "An extended attention mechanism for scene text recognition," *Expert Systems with Applications*, vol. 203, p. 117377, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417422007278

[24] F. Zelic and A. Sable, "Ocr with tesseract in python with pytesseract and opencv," 2023. [Online]. Available: https://nanonets.com/blog/ocr-with-tesseract/

[25] Tensorway. (2023) What is optical character recognition (ocr): Its working, limitations, and alternatives. [Online]. Available: https://www.tensorway.com/post/optical-character-recognition

[26] Konfuzio. (2023) Ocr technology: applications and challenges. [Online]. Available: https://konfuzio.com/en/ocr/#vorteile-und-herausforderungen-von-ocr

[27] N. S. Rani, B. N. B J, K. S. K, and S. A, "Binarization of degraded photographed document images- a variational denoising auto encoder," in *2021 Third International Conference on Inventive Research in Computing Applications (ICIRCA)*, 2021, pp. 119–124.

[28] H. Michalak and K. Okarma, "Improvement of image binarization methods using image preprocessing with local entropy filtering for alphanumerical character recognition purposes," *Entropy*, vol. 21, no. 6, 2019. [Online]. Available: https://www.mdpi.com/1099-4300/21/6/562

[29] S. Klink, A. Dengel, and T. Kieninger, "Document structure analysis based on layout and textual features," vol. 99, 09 2000.

[30] T. Breuel, "High performance document layout analysis," 05 2003.

[31] M. Zampieri, "Chapter 8 - automatic language identification," in *Working with Text*, ser. Chandos Information Professional Series, E. L. Tonkin and G. J. Tourte, Eds. Chandos Publishing, 2016, pp. 189–208. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9781843347491000081

[32] E. O. Systems. (2023, Aug) What are the challenges and limitations of ocr technology? [Online]. Available: https://www.electronicofficesystems.com/2023/08/02/what-are-the-challenges-and-limitations-of-ocr-technology/

[33] G. for Geeks. (2022, Feb) Advantages and disadvantages of optical character reader (ocr). [Online]. Available: https://www.geeksforgeeks.org/advantages-and-disadvantages-of-optical-character-reader-ocr/

[34] R. Agarwal. (2023, Sep) Deep learning based ocr for text in the wild. [Online]. Available: https://nanonets.com/blog/deep-learning-ocr/

[35] L. Jamieson, C. F. Moreno-García, and E. Elyan, "Deep learning for text detection and recognition in complex engineering diagrams," *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–7, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:221607204

[36] O. Suissa, A. Elmalech, and M. Zhitomirsky-Geffet, "Text analysis using deep neural networks in digital humanities and information science," 2023.