Cairo University
Faculty of Engineering
Department of Computer Engineering

Pocket Lens

A Graduation Project Report Submitted

to

Faculty of Engineering, Cairo University

in Partial Fulfillment of the requirements of the degree

of

Bachelor of Science in Computer Engineering.

Presented by

Ahmed Mohamed Ismail
Moaz Mohamed El Sehbini
Mostafa Ashraf Ahmed
Nader Youhanna Adib Khalil

Supervised by

Assoc. Prof. Dr. Mona Farouk

12/06/2023

Communications and Computer (CCEC)

<table>
<tr><td>Project Code</td><td colspan="2">GP-N481-2010</td></tr>
<tr><td>Project Title</td><td colspan="2">Pocket Lens</td></tr>
<tr><td>Keywords</td><td colspan="2">Machine Learning, Computer Vision, Convolutional Neural Networks, Object Detection</td></tr>
<tr><td rowspan="2">Students</td><td>Name: Ahmed Mohamed Ismail
 
E-mail:
ahmedmoh123@hotmail.com
Phone: +201028300083
Address: Shobra, Cairo</td><td>Name: Moaz Mohamed Elsherbini
 
E-mail: moaz5657@gmail.com
Phone: +201018711749
Address: 6th October City, Giza</td></tr>
<tr><td>Name: Mostafa Ashraf Ahmed Kamal
 
E-mail:
moustafa.achraf@hotmail.com
Phone: +201003993985
Address: Nasr City, Cairo</td><td>Name: Nader Youhanna Adib Khalil
 
E-mail:
naderyouhanna@gmail.com
Phone: +201285003523
Address: Manial, Cairo</td></tr>
<tr><td rowspan="2">Supervisor</td><td>Name: Assoc. Prof. Dr. Mona Farouk</td><td>E-mail: mona_farouk@eng.cu.edu.eg</td></tr>
<tr><td>Signature:</td><td>Phone: +201005042029</td></tr>
<tr><td>Project Summary</td><td colspan="2">An application that helps visually impaired individuals navigate the world by describing objects in scenes with their distances, describing clothes, identifying products, detecting faces, reading texts, and recognizing people, emotions, and currencies.</td></tr>
</table>

Abstract

Even though many mobile devices today include accessibility features available for visually impaired and blind users, many of these users are reluctant to use them. This is because either the features are not very beneficial for the user, or the interface is mainly designed for sighted people. The latter is caused by the fact that the main input and output methods on mobile devices are tactile or visual in nature. However, in recent years, there have been many innovative applications that assist VIB users in navigating their environment. Programmers have made use of technological advances regarding gyroscope sensors and vibration feedback to make communication possible.

The proposed system relies on input images and videos provided by the user's device camera to allow daily life navigation without the need to use such sensors. It makes communication between VIB users and their devices possible using speech/text conversion techniques.

The approach that is followed to solve this problem is to use artificial intelligence to analyze images captured by the device's camera and provide feedback to the user through speech synthesis. The output of this project is a mobile app that can run on both Android and iOS devices, and that can be customized according to the user's preferences and needs.

The summary of testing results shows that the app is effective, accurate, and reliable in performing the intended functions and that it has a positive impact on the user's independence and quality of life.

الملخص

على الرغم من أن العديد من الأجهزة المحمولة اليوم تتضمن ميزات للمستخدمين المكفوفين وضعاف البصر، إلا أن العديد من هؤلاء المستخدمين يترددون في استخدامها. هذا لأن الميزات ليست مفيدة جدًا للمستخدم أو أن الواجهة مصممة بشكل رئيسي للأشخاص ذوي البصر السليم. مشكلة الواجهات المصممة هي كون طرق الإدخال والإخراج الرئيسية على الأجهزة المحمولة تكون عن طريق اللمس واستخدام حاسة البصر. ومع ذلك، في السنوات الأخيرة، ظهرت العديد من التطبيقات المبتكرة التي تساعد المستخدمين المكفوفين وضعاف البصر في التنقل في بيئتهم. قام المبرمجون باستخدام التطورات التكنولوجية المتعلقة بحساسات الجيروسكوب والاهتزاز لجعل التواصل ممكنًا. النظام المقترح يعتمد على صور وفيديوهات إدخال يقدمها كاميرا جهاز المستخدم للسماح بالتنقل في الحياة اليومية دون الحاجة إلى استخدام مثل هذه الحساسات. يجعل التواصل بين مستخدمي VIB وأجهزتهم ممكنا باستخدام تقنيات التحويل من/إلى نص/كلام.

الطريقة التي يتبعها النظام لحل هذه المشكلة هي استخدام الذكاء الاصطناعي لتحليل الصور التي يلتقطها كاميرا جهاز وتقديم ملاحظات للمستخدم من خلال توليف الكلام. ناتج هذا المشروع هو تطبيق جوال يعمل على كلاً من أجهزة Android وiOS، والذي يمكن تخصيصه وفقًا لتفضيلات واحتياجات المستخدم. تظهر ملخص نتائج اختبارات أن التطبيق فعال ودقيق وموثوق في أداء الوظائف المقصودة، وأن لديه تأثير إيجابي على استقلالية المستخدم وجودة حياته.

ACKNOWLEDGMENT

We would like to express our sincere gratitude to Allah for giving us the opportunity and the strength to complete this graduation project.

We would like to thank our families as well. Without their support, we would not have been able to achieve what we did.

We would also like to thank Dr. Mona Farouk, our supervisor, and mentor, for her invaluable guidance, feedback, and encouragement throughout this journey. She has been a source of inspiration and motivation for us, and we have learned a lot from her expertise and experience. We are truly grateful for her support and kindness.

Ahmed, Moaz, Mostafa and Nader

Table Of Contents

List of Figures

List of Tables

List of Abbreviation

Abbreviation	Definition
AI	Artificial intelligence
CV	Computer Vision
GPU	Graphical Processing Unit
HOG	Histogram of Oriented Gradient
RPI	Retail Product Identifier
TF-IDF	Term Frequency – Inverse Document Frequency
UI	User Interface
UX	User Experience
VIB	Visually impaired and blind
VSCode	Visual Studio Code
WHO	World Health Organization
YOLO	You Only Look Once

Contacts

Team Members

Name	Email	Phone Number
Ahmed Mohamed Ismail	ahmedmoh123@hotmail.com	+2 01028300083
Moaz Mohamed El Sherbini	moaz5657@gmail.com	+2 01018711749
Mostafa Ashraf Ahmed	moustafa.achraf@hotmail.com	+2 01003993985
Nader Youhanna Adib	naderyouhanna@gmail.com	+2 01285003523

Supervisor

Name	Email	Number
Dr. Mona Farouk	mona_farouk@eng.cu.edu.eg	+2 01005042029

This page is left intentionally empty.

Chapter 1: Introduction

According to the WHO, around 2 billion people are visually impaired or blind. This is not a minority. Nevertheless, very little has been done to help them throughout their day. Mobile phones offer accessibility features for them, but these features are not enough for day-to-day activities.

The proposed system offers a mobile application that uses AI to help VIB people complete their daily tasks. It captures images from the user's camera as input and gives the user feedback through a text-to-speech module.

1.1 Motivation and Justification

VIB users are often put at a disadvantage regarding their visually able peers. Technological advancements have always been concerned with providing better and easier-to-use solutions. These efforts have been largely directed toward the use of sensors, which in many are not available to every user.

Moreover, many of the applications that can be found in the market are not particularly easy to use. They often require some degree of tactile interaction, which VIB users will most probably not be able to provide. Some of these applications are designed to be used by sighted people alongside VIB users, which can come as impractical.

The above-mentioned reasons led us to consider using AI and Machine Learning techniques to create a mobile application that can serve as an assistant to VIB people. We will be addressing these previous problems by rendering the contact between the application and the VIB user purely vocal as much as the desired features allow for it. In other words, the user will communicate with the chatbot through speech.

1.2 The Essential Question

The essential question is how to use AI and Machine Learning techniques to create a mobile application that can serve as an assistant to visually impaired and blind (VIB) people. This is relevant to the Vision and Mission of the Faculty of Engineering at Cairo University as it aligns with their goal of using technological advancements to provide better and easier-to-use solutions for everyone. The proposed system aims to address previous problems by rendering the contact between the application and the VIB user purely vocal, allowing for easier communication and interaction.

1.3 Project Objectives and Problem Definition

The problem being addressed is the disadvantage faced by visually impaired and blind (VIB) users in comparison to their visually able peers. The objective of the project is to use AI and Machine Learning techniques to create a mobile application that can serve as an assistant to VIB people. The application aims to address previous problems by rendering the contact between the application and the VIB user purely vocal, allowing for easier communication and interaction.

1.4 Project Outcomes

The outcome of the project would be a mobile application that uses AI and Machine Learning techniques to serve as an assistant to visually impaired and blind (VIB) people.

1.5 Document Organization

This document is organized into six chapters, starting with this introduction which provides an overview of the project.

The second chapter is the Visibility Study, which examines the need for the project, its potential impact, its target customers, and the market survey.

The third chapter is the Literature Survey, which reviews previous work done in the field and identifies gaps that the project aims to address.

The fourth chapter is System Design and Architecture, which outlines the technical details of the project including its components and how they interact.

The fifth chapter is System Testing and Verification, which discusses the process used to ensure that the system performs as intended and meets the requirements.

Finally, the document ends with the Conclusion and Future Work chapter, which summarizes the project and suggests areas for further development and improvement.

Chapter 2: Market Visibility Study

The project market is an innovative virtual assistant designed specifically for visually impaired and blind individuals. This cutting-edge technology aims to improve the quality of life for those with visual impairments by providing them with a tool that can assist them in their daily lives. With its advanced features and user-friendly interface, the project market is set to revolutionize the way visually impaired and blind people interact with the world around them.

2.1 Targeted Customers

The target customers for our mobile application are VIB individuals who are looking for a more accessible and intuitive way to interact with their smartphones. Our application aims to address the disadvantage that VIB users often face in comparison to their visually able peers. The application's purely vocal interaction allows for a more natural and convenient way for VIB users to communicate with their phones. Our target customers are those who seek a mobile application that is easy to use, reliable, and tailored to their specific needs, allowing them to access the same features and functionalities as their sighted counterparts.

2.2 Market Survey

In this section, we will list the competitive products for our application. We will explore similar commercial tools and platforms and discuss them. A subsection will be dedicated to each one of them.

2.2.1 Smart Glasses of Envision

Envision smart glasses are designed to help VIB individuals read. They were built on the enterprise edition of Google Glasses and rely heavily on Artificial Intelligence. The glasses aim to articulate everyday visual information into speech. The features of the glasses include scanning text, scene description, light detection, cash recognition, color detection, finding people, finding objects, teaching face and exploring. Our product is different from Envision’s glasses in the measure that it is an application, which means no hardware purchase is required by our customers. Moreover, no hardware maintenance is required. This cuts production costs but has a negative impact on performance since we have no access to extra sensors.

2.2.2 Google Lookout

Google's Lookout is an application that can help people identify food labels as well as find objects in a room. It can also scan documents, money, and products.

Lookout uses computer vision and machine learning technology to assist people with low vision or blindness to get things done faster and more easily. Lookout is available for free for Android devices on the Play Store. Using their phone's camera, Lookout makes it easier to get more information about the world around people and do daily tasks more efficiently like sorting mail, putting away groceries, and more. After identifying objects in the scene, the application provides audio feedback about what it detects in the environment. The user can also customize the application's settings to receive specific types of feedback and adjust the volume and speed of the audio output. The application is also integrated with TalkBack, Google's screen-reading software, which enhances its accessibility for VIB individuals.

However, one potential limitation of Google Lookout is that its performance varies depending on lighting conditions, camera quality, and user proficiency. It also requires a stable internet connection. Moreover, the application is only available on Android devices, which may limit its accessibility to users who prefer other operating systems.

2.2.3 Be My Eyes

Be My Eyes is another application that is intended to help VIB individuals to navigate the world. It connects visually impaired users with sighted volunteers. The volunteers then help the user get around via a live chat function. It also aims to be integrated in the future with OpenAI's ChatGPT-4 and an AI-powered volunteer that will provide instantaneous identification, interpretation, and conversational visual assistance for a wide variety of tasks.

To use the application, a visually impaired user can request assistance through a video call, and a sighted volunteer will answer the call and provide assistance in real-time by describing the visual surroundings or helping with a task. The volunteers are trained to assist in a variety of areas, such as reading labels, identifying colors, or navigating unfamiliar environments.

The application also offers a specialized feature called "Specialized Help," which connects users with representatives from various partner organizations, such as

Microsoft, Google, and the American Diabetes Association, who can assist with specific issues related to their products or services.

One of the strengths of Be My Eyes is its ease of use and accessibility, as the application is designed to be simple and intuitive. It also provides a valuable service to VIB individuals by leveraging the power of technology and human connection.

However, one potential limitation of Be My Eyes is its reliance on volunteers, which can result in inconsistent availability and varying levels of expertise.

2.2.4 Microsoft Soundscape

Microsoft Soundscape, developed by the Enable Group in Microsoft Research, is a groundbreaking voice-based navigation app that empowers visually impaired individuals to explore the world independently. Utilizing GPS, compass, and audio feedback, the app delivers a 3D sound map, allowing users to navigate their surroundings with the help of audio cues and markers. While it provides an immersive experience for VIB individuals, its accuracy may be compromised in crowded or complex environments with overlapping audio. Additionally, a stable internet connection and an iOS device with GPS and compass are required to use the app.

However, Microsoft Soundscape is not as accurate in crowded or complex environments where multiple sounds and audio may overlap. It also requires a stable internet connection and a smartphone with a GPS and compass. Furthermore, it is only available on iOS devices.

2.2.5 Facing Emotions

Facing Emotions is an application developed by Huawei. It identifies the 7 basic human emotions of irritation, contempt, sorrow, fear, anger, surprise, and happiness. The app turns those emotions into unique sounds to help the visually impaired learn how the person on the other side of the conversation is feeling. The application is designed to help users improve their emotional intelligence and communication skills by providing real-time feedback on their facial expressions. However, the application is only available for Huawei smartphones, which may limit its accessibility to users who prefer other brands or operating systems.

2.3 Business Case and Financial Analysis

The development of a mobile application for VIB individuals presents a significant business opportunity in a growing market. According to the World Health Organization, there are approximately 285 million visually impaired individuals worldwide, with 39 million of them being blind. The demand for assistive technology and solutions is expected to increase as the population ages and the prevalence of visual impairments rises. The development of an accessible and intuitive mobile application can meet this demand and provide a valuable service to visually impaired individuals.

The financial analysis of the project will depend on various factors such as development costs, marketing expenses, and revenue streams. The development costs will include the expenses related to the software development, designing, and testing of the application. The marketing expenses will include the cost of advertising, promotion, and distribution of the application. The revenue streams can be generated through various sources such as in-app purchases, subscriptions, or advertisements. However, our project is non-lucrative, and we do not aim to generate revenue through it. Instead, our focus is on addressing the needs and challenges of VIB individuals by providing them with an accessible and intuitive way to interact with their smartphones. This approach aligns with the principles of social responsibility and inclusivity and can help to improve the quality of life of VIB individuals.

2.4 SWOT Analysis

Table 2-1 SWOT Analysis

Strengths	Weaknesses
• AI chatbot • 24/7 availability • Privacy • Internet access not needed (maps can be saved offline) • Understand other people better by knowing their facial emoticons	• AI functions are not 100% accurate hence can identify wrong currencies or emotions. • Lack of empathetic human care • Expertise need
Opportunities	**Threats**
• Improving humans' life • Convenience • Spreading awareness	• Risk of over-depending on the technology • Data security

Chapter 3: Literature Survey

Computer Vision (CV) is a field of AI that is interested in enabling machines to interpret, process, and understand visual data. Since our project depends significantly on CV, the first part of this section will be dedicated to explaining some key concepts of AI that are pivotal for understanding this section and the project in general.

Some of the key techniques used in computer vision include machine learning, deep learning, image processing, pattern recognition, and computer graphics.

Image processing techniques include image filtering, which is used to enhance or suppress some features of an image, blur an image, or highlight some specific features.

In a CV pipeline, most often comes the next feature extraction, where the algorithm identifies and extracts important features from an image. These are usually corners, edges, and textures.

Deep Learning, which is a subset of machine learning, is also one of the prominent techniques used in computer vision. Convolutional Neural Networks, or CNNs for short, are a type of neural network that uses the convolution operation.

3.1 Comparative Study of Previous Work

In this subsection, we will conduct a comparative study of previous work that has been done in the field. This will help us to understand the existing research and identify gaps that we can address in our study.

To begin with, we will review the literature on the topic and identify the key studies that have been conducted in this area.

3.1.1 Scene Descriptor

Diwan, Anirudh, & Tembhurne, 8 August 2022 [1] highlights the significant advancements in object detection and related tasks using deep learning models. Object detectors are classified into two categories: two-stage and single-stage detectors, with each having its strengths and weaknesses. Two-stage detectors excel in detection accuracy but suffer from longer inference times, while single-stage detectors offer faster

inferences at the cost of slightly lower accuracy. However, recent developments, particularly the YOLO (You Only Look Once) architecture and its successors, have significantly improved detection accuracy while still maintaining faster inference times. This review paper provides an overview of single-stage object detectors, specifically focusing on YOLOs, their regression formulation, architectural advancements, and performance statistics. A comparison is made between two-stage and single-stage detectors, different versions of YOLOs, and their respective applications, along with insights into future research directions.

Miangoleh, Dille, Mai, Paris, & Aksoy, 2021 [2] shows that Neural networks have exhibited impressive capabilities in estimating depth from a single image. However, the resulting depth maps have limitations in terms of resolution and fine-grained details. In our approach, we analyzed to understand the impact of input resolution and scene structure on depth estimation performance. We discovered a trade-off between consistent scene structure and high-frequency details and leveraged this duality using a straightforward depth merging network. Our method involves a double estimation technique to enhance overall depth estimation across the entire image and a patch selection method to incorporate local details into the final output. By merging estimations at different resolutions and considering changing contexts, wecontext rate multi-megapixel depth maps with exceptional detail using a pre-trained model.

3.1.2 Clothes Description

Ge, Zhang, Wang, Tang, & Luo, 2019 [3] talk about DeepFashion which has been a valuable benchmark for understanding fashion images, but it has limitations such as only one clothing item per image, sparse landmarks, and the absence of per-pixel masks. To address these issues and bridge the gap with real-world scenarios, we introduce DeepFashion2. This versatile benchmark includes four tasks: clothes detection, pose estimation, segmentation, and retrieval. With over 800,000 clothing items, DeepFashion2 provides rich annotations including style, scale, viewpoint, occlusion, bounding box, dense landmarks, and masks. Additionally, it offers a substantial number of commercial-consumer clothes pairs. To facilitate research, we present a strong baseline called Match RCNN, which leverages Mask R-CNN for end-to-end solutions to the aforementioned tasks

Xin, Zhang, Zhang, & Wu, 2017 [4] show that Fabric analysis often requires accurate color texture classification, especially in textile manufacturing. In this study, a novel artificial intelligence approach combining a dual-side co-occurrence matrix and a back propagation neural network was proposed for color texture classification. The method demonstrated improved classification results for yarn-dyed woven fabric compared to

traditional single-side co-occurrence matrices. A laboratory dual-side imaging system was used to capture upper-side and lower-side fabric images. The dual-side co-occurrence matrix was generated, and four texture features were extracted to evaluate fabric texture characteristics. A well-trained back propagation neural network utilized these features as input vectors and predicted the color texture type. The performance of the dual-side co-occurrence matrix approach was compared to the single-side matrix method, and the results showed that the proposed artificial intelligence system achieved better classification accuracy.

3.1.3 Face Detection

Viola & Jones [5] propose a method for object detection in images using Haar-like features and the AdaBoost algorithm. The authors introduced Haar-like features, which are rectangular features that measure the difference in pixel intensities between adjacent regions in an image. AdaBoost is used to select the most discriminative features for a given object, and a cascade of classifiers is trained using these features. The authors achieved high detection rates with low false positives on face detection tasks and demonstrated the speed of their method in real-time video. Their approach has since been widely adopted in various applications.

3.1.4 Face Recognition

We have read one paper about face recognition. Turk & Pentland [6] introduce the Eigenfaces algorithm for Face Recognition. The paper uses a linear combination of eigenvectors of the face image matrix to represent and classify faces. The method involves constructing a subspace of the face images using principal component analysis (PCA) and projecting new faces onto this subspace to obtain their eigenface representations. The eigenfaces can then be used to classify the faces by comparing their distances to the mean face of each individual. To classify a new face image, it is projected onto the subspace spanned by the eigenfaces, and its representation is compared to the representations of known faces using a distance metric. The distance metric used in the paper is the Euclidean distance. The paper shows that this method achieves high recognition rates even with a small number of training images and can be used for real-time face recognition applications.

3.1.5 Emotion Detection

Kim, Joo, & Park [7] proposed an emotion detection algorithm using a frontal facial image. There are three stages: image processing, facial feature extraction, and emotion detection. In the image processing stage, the face region is extracted by using fuzzy color filter, virtual face model, and histogram analysis method. The features for emotion detection are extracted from facial components in facial feature extraction stage. In emotion detection stage, the fuzzy classifier is adopted to recognize emotion from extracted features.

3.1.6 Product Identifier

Wei, Tran, Xu, Kang, & Springer [8] discusses challenges and techniques associated with using deep learning for product recognition in the retail industry. Challenges include variability in appearance and working with large datasets. Techniques presented include data augmentation, transfer learning, and ensembling. Architecture selection and hyperparameter optimization are also discussed. The paper provides valuable insights for researchers and practitioners in this field.

3.1.7 Apparel Recommender

Yang, Yuan, & Tian [9] focuses on recognizing clothing patterns in four categories (plaid, striped, pattern less, and irregular) and identifies 11 clothing colors. A camera mounted upon a pair of sunglasses is used to capture clothing images. The clothing patterns and colors are described to blind users verbally. This system can be controlled by speech input through microphone.

3.1.8 Currency Recognizer

In his paper, Jamkhandikar D [10] propose an Indian Currency Recognition System designed to identify and classify Indian banknotes. The authors utilize various image processing techniques, including edge detection, segmentation, and feature extraction, to analyze images of Indian currency notes. Machine learning techniques, such as (SVM), (ANN), and (k-NN), are employed for classification, enabling the system to accurately recognize different denominations of Indian banknotes. The study demonstrates the potential of combining image processing and machine learning approaches in creating an effective currency recognition system.

In the paper the authors, Zhang Q [11] present a modern approach to currency recognition by employing deep learning techniques. Leveraging CNN, a powerful tool in image classification and object recognition tasks, the researchers developed a robust system capable of recognizing and classifying various banknote denominations. The study highlights the effectiveness of deep learning methods in currency recognition, outperforming traditional image processing and machine learning techniques. This research contributes to the growing body of literature on automated currency recognition systems.

3.1.9 Text Reader

In their paper, Aqab S, Tariq M [12] the authors explore the application of AI and image processing techniques for recognizing and analyzing handwritten text. The study focuses on preprocessing handwritten images, extracting relevant features, and employing neural networks for classification and recognition. By combining image processing methods such as binarization, noise removal, and segmentation with the power of ANNs, the authors demonstrate the potential of AI-based approaches to accurately recognize and interpret handwritten characters and words. This research contributes to the field of handwriting recognition and has significant implications for various applications, including OCR systems, document analysis, and natural language processing.

A paper written by Beohar D and Rasool A [13] investigate the application of advanced deep learning techniques for recognizing handwritten digits from the widely used MNIST dataset. The study compares the performance of state-of-the-art ANNs and CNNs to identify the most effective approach for this task. The authors demonstrate the superior performance of CNNs, which leverage their ability to learn hierarchical features and spatial invariance, resulting in higher accuracy and robustness compared to traditional ANNs. This research not only contributes to the understanding of deep learning techniques in the context of handwritten digit recognition but also highlights the potential of CNNs for various image recognition and classification tasks.

3.2 Implemented Approach

For the Text Reader we used EMNIST dataset to train CNN model trained and extract text from any Image. This method was efficient since the dataset contains a wide range of letters which helps the model recognize any font.

While in Currency Recognition it was implemented using KNN since currency have different texture and hence, we can extract distinct features from them.

As for the face detection, the viola-jones approach was used. The fact that the algorithm had such a high true positive rate while achieving real time performance made it the best option.

The face recognition module was implemented using the Eigenfaces technique as it is one of the most popular and effective methods for face recognition and has been shown to achieve high accuracies. It also presents a low computational cost, making it suitable for real-time applications.

Regarding emotion detection, the system used was similar to what was used in regard to facial landmark extraction. However, for classification, random forest classification is used because, generally, a random forest classifier is more suitable for data with many input features.

In the product identifier module, we used the YOLO object detection algorithm as it is fast and accurate. It can detect multiple objects in an image and is customizable, making it suitable for identifying specific products. It is optimized for speed and can detect objects in real-time, making it ideal for applications that require quick identification of products.

We have also used YOLO in the Scene Descriptor module as it has achieved state-of-the-art accuracy on object detection benchmarks, meaning it can reliably detect different objects in images. In addition, we used depth estimation techniques using Midas and triangulation to estimate the distance of objects detected in the scene.

As for the apparel recommender we have decided to use the same approach as the one used in the paper reviewed. However, we have reduced the number of features for simplicity. Moreover, we added some rules that the recommender should follow in giving suggestions.

In the Clothes Descriptor module, we trained a model using Mask R-CNN for object detection and instant segmentation. Then the segmented results are fed to texture analysis model which has been trained with traditional machine learning techniques, mainly random forest to detect clothes textures.

Chapter 4: System Design and Architecture

During the development of our project, we followed a structured approach based on software engineering principles to ensure that we built a reliable and effective solution. We began by defining the requirements and specifications of the app, which included trying to understand the challenges faced by VIB individuals. Based on this step, we selected our modules which are: Scene Descriptor, Clothes Descriptor, Face Detector, Face Recognizer, Emotion Detector, Currency Recognizer and Text Reader. We then implemented the solution using reliable and efficient coding practices, such as using project flow control tools like git to minimize errors and ensure high-quality code. We also conducted rigorous quality assurance processes to ensure that the application functioned as intended and was robust and secure. By following this software engineering approach, we were able to build a reliable and effective solution that can make a positive impact on people's lives.

4.1 Overview and Assumptions

The system is designed to be a mobile application that is built using different architectures and models for each module including YOLO, Random Forests, PCA and Logistic Regression.

We have taken the freedom of assuming that the VIB user is able to download the application or that it is already installed on their device. We have also assumed that they have enough means of opening the application. However, once the application is opened and running, it can be entirely controlled by voice commands.

4.2 System Architecture

The main processing code for the modules is executed on a server while the application, built using the Flutter framework, is installed on the user's device. The application sends requests to the server, specifying the image to be processed and the desired functionality. The server then processes the image using one of the available modules and returns a result string, typically containing a description of the image.

4.2.1 Block Diagram

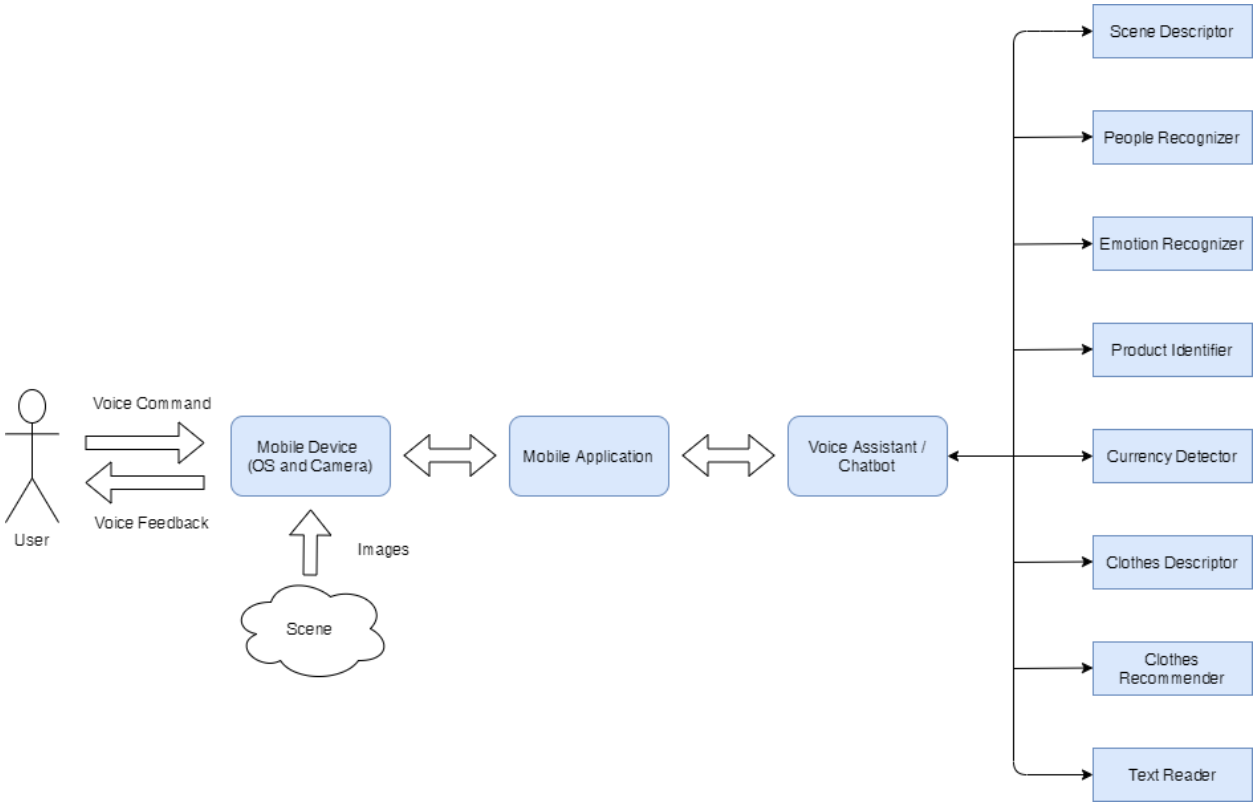


Figure 4-1 – System Block Diagram

4.3 Scene Descriptor

The Scene Description Module is a critical component of the mobile application designed to cater to the needs of visually impaired individuals. This module plays a vital role in enhancing their understanding of the surrounding environment by providing accurate and meaningful descriptions of the scene captured through the mobile camera. By leveraging cutting-edge computer vision techniques, the module enables them to gain insights into their surroundings and navigate them more confidently.

The Scene Description Module incorporates a virtual assistant that recites the detected objects back to the user in the form of speech. Additionally, the module estimates the approximate depth of the detected objects, indicating whether they are closer or further than 1 meter from the user's position.

The Scene Description Module follows a well-defined pipeline to obtain accurate results. Next, we shall discuss this pipeline in depth to better understand how this module works.

4.3.1 Functional Description

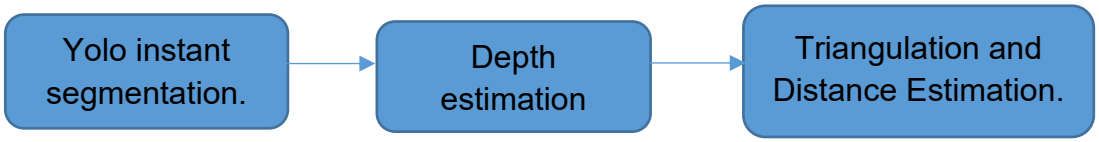


Figure 4-2 Scene Descriptor Components

1. YOLO instant segmentation is a powerful object detection algorithm used in the Scene Description Module, accurately identifying and segmenting objects in the captured image.

2. Depth estimation, employing techniques like MIDAS, provides depth information about objects in the scene, enabling visually impaired individuals to perceive the spatial layout of their surroundings.

3. Triangulation and distance estimation utilize a formula combining real-life object width, focal length, object pixel width, and depth value to estimate the distance between the user and objects in the scene, aiding spatial understanding.

Together, these components of the Scene Description Module—YOLO instant segmentation, depth estimation, and triangulation with distance estimation—enhance object recognition, depth perception, and spatial awareness for visually impaired users.

4.3.2 Modular Decomposition

4.3.2.1 COCO Dataset

The COCO (Common Objects in Context) dataset is a widely used benchmark dataset in computer vision research, including object detection and scene understanding tasks. It serves as a valuable resource for training and evaluating deep learning models due to its large-scale and diverse collection of images.

1. Dataset Description:

The COCO dataset consists of over 200,000 images with more than 80 object categories, making it a rich source for training object detection models. The images in the dataset are carefully annotated with pixel-level object segmentation masks, bounding box annotations, and class labels. This comprehensive annotation allows for accurate evaluation and analysis of object detection algorithms.

2. Training with COCO dataset

During training, the deep learning model is exposed to the images and their corresponding annotations from the COCO dataset. The model learns to detect and classify objects by optimizing the model's parameters using gradient-based optimization algorithms such as stochastic gradient descent (SGD) or Adam.
To facilitate effective training, data augmentation techniques are commonly employed. These techniques involve applying various transformations to the training images, such as rotation, scaling, cropping, and flipping. Data augmentation helps enhance the model's ability to generalize and improves its performance on unseen images.

3. Advantages and Limitations

The COCO dataset offers several advantages for training object detection models. Its large-scale nature provides a diverse range of images and object categories, allowing models to generalize better in real-world scenarios. The detailed annotations, including segmentation masks and bounding boxes, enable precise evaluation and analysis of object detection algorithms.

However, the COCO dataset also has certain limitations. While it covers a broad range of object categories, some categories may have fewer instances compared to others. This

class imbalance can affect the model's performance, particularly for less-represented categories. Additionally, the dataset may not capture all possible object variations and occlusions present in real-world scenes, leading to challenges when deploying the trained models in complex environments.

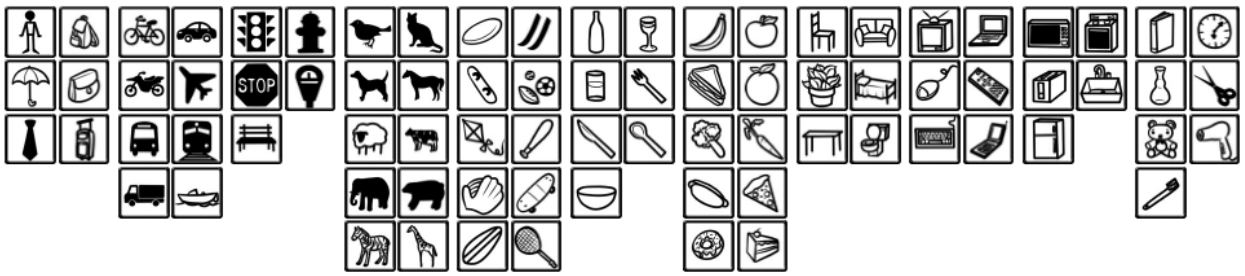


Figure 4-3: COCO Dataset Classes Images

Table 4-1: COCO Dataset Classes Names

person	fire hydrant	elephant	skis	wine glass	broccoli	dining table	toaster
bicycle	stop sign	bear	snowboard	cup	carrot	toilet	sink
car	parking meter	zebra	sports ball	fork	hot dog	tv	refrigerator
motorcycle	bench	giraffe	kite	knife	pizza	laptop	book
airplane	bird	backpack	baseball bat	spoon	donut	mouse	clock
bus	cat	umbrella	baseball glove	bowl	cake	remote	vase
train	dog	handbag	skateboard	banana	chair	keyboard	scissors
truck	horse	tie	surfboard	apple	couch	cell phone	teddy bear
boat	sheep	suitcase	tennis racket	sandwich	potted plant	microwave	hair drier
traffic light	cow	frisbee	bottle	orange	bed	oven	toothbrush

4.3.2.2 Object Detection and Instant Segmentation using Yolov8.

Object detection is a fundamental task in computer vision, and YOLOv8 (You Only Look Once) is a state-of-the-art algorithm that has demonstrated remarkable performance in real-time object detection. YOLOv8 operates by dividing the input image into a grid and simultaneously predicting each grid cell's bounding boxes and class probabilities. This approach allows YOLOv8 to achieve both accuracy and efficiency.

1. Architecture and Workflow

The YOLOv8 architecture consists of a deep convolutional neural network (CNN) that is composed of multiple convolutional layers, followed by fully connected layers. CNN

extracts high-level features from the input image, enabling the network to learn discriminative representations for various object categories.

The YOLOv8 network follows a single-shot detection paradigm, meaning that it performs detection in a single pass through the network. The input image is divided into a predetermined number of grid cells, typically 13x13 or 19x19, depending on the network configuration. Each grid cell is responsible for predicting bounding boxes and class probabilities for objects present within its spatial region.

2. Bounding Box Prediction

For each grid cell, YOLOv8 predicts multiple bounding boxes. Each bounding box is represented by a set of parameters, including the coordinates of the box's top-left corner, its width, height, and confidence score. The confidence score indicates the likelihood of the bounding box containing an object.

To obtain accurate bounding box predictions, YOLOv8 uses anchor boxes. Anchor boxes are pre-defined bounding box shapes of various aspect ratios and scales. These anchor boxes are learned during the training phase and serve as reference templates for predicting the final bounding box coordinates. By utilizing anchor boxes, YOLOv8 can effectively handle objects of different shapes and sizes.

Figure 4-4: Scene Descriptor Yolo Bounding Boxes

3. Class Probability Prediction

Alongside the bounding box predictions, YOLOv8 also predicts class probabilities for each grid cell. The class probabilities represent the likelihood of an object belonging to a specific class or category. YOLOv8 typically employs a SoftMax activation function to normalize the class probabilities, ensuring that they sum up to one for each grid cell.

4. Instant Segmentation

Within the YOLOv8 object detection algorithm, instant segmentation is achieved through the integration of a segmentation branch into the network architecture. This segmentation branch allows YOLOv8 to generate pixel-wise segmentation masks for the detected objects, enabling precise delineation of the object boundaries within the scene.

To accomplish instant segmentation, YOLOv8 incorporates additional convolutional layers in parallel with the object detection layers. These layers are designed to capture fine-grained details and spatial information required for accurate segmentation.

During the forward pass of YOLOv8, both the object detection branch and the segmentation branch are simultaneously activated. The object detection branch predicts bounding boxes and class probabilities, while the segmentation branch predicts the segmentation masks for each detected object.

The segmentation masks are generated by assigning a pixel-wise probability score to each pixel within the bounding box of the detected object. These scores indicate the likelihood of each pixel belonging to the object class.

Figure 4-5: YOLO segmented detected objects

5. Non-Maxima Suppression

After obtaining the bounding box predictions and class probabilities for each grid cell, a post-processing step called non-maximum suppression (NMS) is applied. NMS filters out redundant and overlapping bounding box predictions to provide a more accurate and concise set of final detections.

6. Training YOLOv8

To train YOLOv8, a large, labeled dataset is required. This dataset contains images with annotated bounding boxes and class labels for the objects of interest. During training, YOLOv8 optimizes its parameters by minimizing a loss function that incorporates the localization loss (bounding box coordinates) and classification loss (class probabilities).

Training YOLOv8 involves a two-step process: pretraining on a large-scale dataset, such as ImageNet, to learn general feature representations, and fine-tuning on the specific object detection task using the labeled dataset. The pretraining step helps YOLOv8 to capture generic visual features, while the fine-tuning step allows it to adapt to the specific object detection requirements.

Overall, YOLOv8 offers a powerful solution.

4.3.2.3 Depth Estimation with Midas

Depth estimation is a crucial task in scene understanding and plays a vital role in the Scene Description Module. MIDAS (Monocular Depth Estimation in Real Time) is a deep learning-based model that utilizes a single RGB image to estimate the depth map, providing valuable information about the spatial layout of objects within the scene.

1. Architecture and Workflow

MIDAS employs a convolutional neural network (CNN) architecture designed to capture both local and global context information from the input image. The network leverages a series of convolutional layers with increasing receptive fields to extract features at multiple scales.

The input RGB image is fed into the MIDAS network, which then processes it through the convolutional layers to obtain a feature representation. This feature representation is then passed through additional layers to predict the corresponding depth map. The depth map produced by MIDAS contains per-pixel depth values, representing the relative distances of objects from the camera.

2. Training MIDAS

Training MIDAS requires a large-scale dataset with RGB-D (RGB and depth) pairs. These datasets provide images along with their corresponding ground truth depth maps.

During training, MIDAS learns to estimate depth maps by minimizing the discrepancy between the predicted depth maps and the ground truth.

The training process involves optimizing the network's parameters using a loss function that quantifies the difference between the predicted depth map and the ground truth depth map. Commonly used loss functions include mean squared error (MSE) or absolute differences between the predicted and ground truth depth values.

To improve the generalization capability of MIDAS, data augmentation techniques are often employed during training. These techniques involve applying transformations such as random scaling, rotation, and flipping to the input images and their corresponding depth maps. Data augmentation helps MIDAS to learn robust depth estimation, allowing it to handle various environmental conditions and scenarios.

3. Depth Map Post-processing

After obtaining the depth map from MIDAS, post-processing techniques can be applied to refine the results and enhance their visual quality. Common post-processing steps include depth map normalization, contrast adjustment, and noise reduction.

Normalization ensures that the depth values are within a specific range, such as between 0 and 1, making the depth map consistent and interpretable. Contrast adjustment techniques can be employed to enhance the visual representation of the depth map, improving its readability for users.

Noise reduction techniques, such as filtering or smoothing algorithms, can be applied to reduce the presence of noise or outliers in the depth map. This helps to improve the accuracy and reliability of the depth estimations.

4. Heat Map Generation

In the context of the Scene Description Module, the depth map obtained from MIDAS is further utilized to generate heat maps for each detected object. These heat maps represent the relative depth and intensity of each object within the scene.

To generate the heat maps, the depth map is segmented based on the bounding boxes obtained from the object detection process (using YOLOv8). Each segmented region corresponds to a specific object detected within the scene. The heat maps are then created by converting the segmented regions of the depth map to grayscale, where the intensity values indicate the relative depth of the object.

Figure 4-7: Heat Map

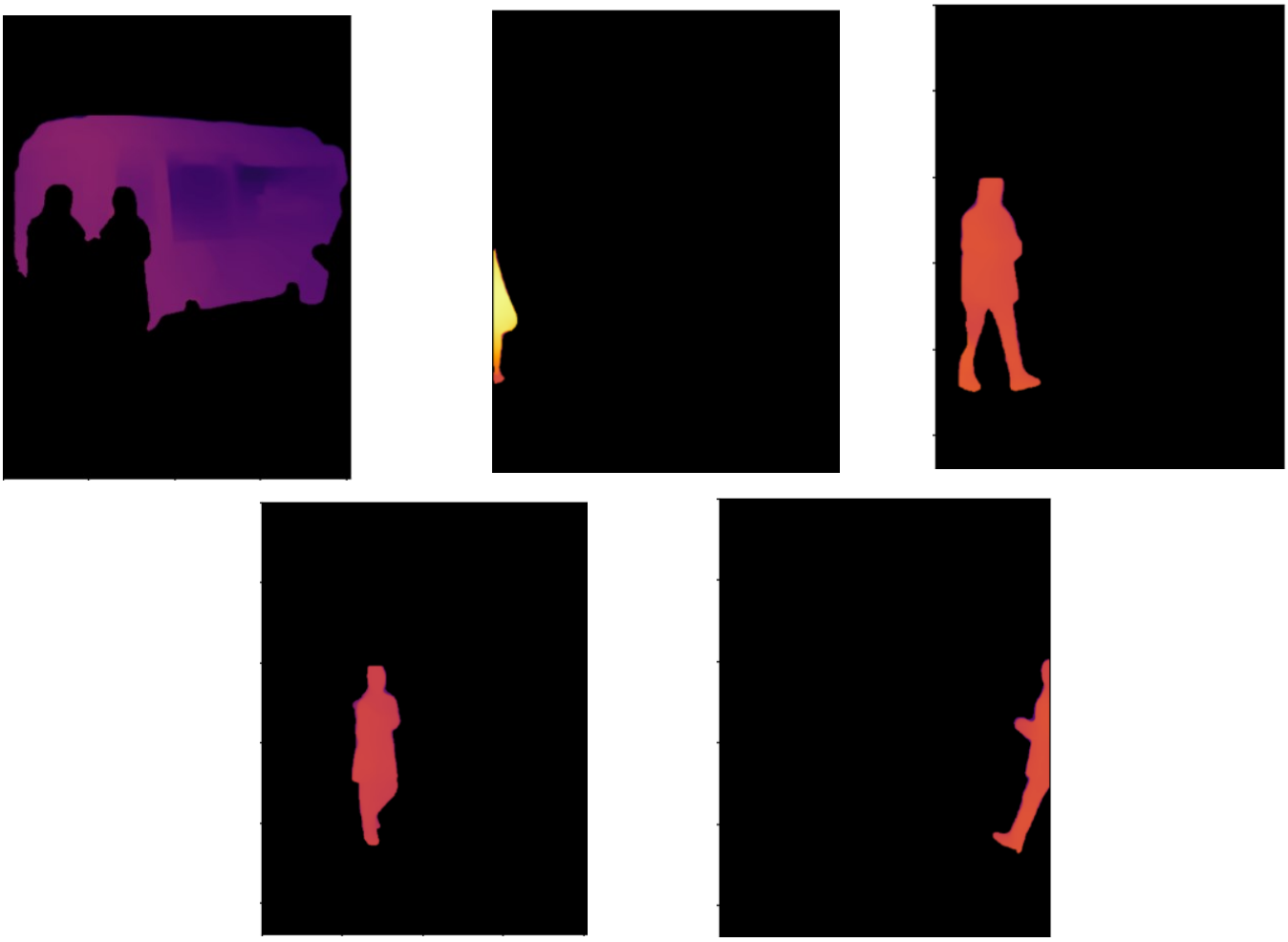


Figure 4-6: Segmented Objects of Heatmap

4.3.2.4 Triangularization and Distance Estimation

Triangulation and distance estimation play a crucial role in the Scene Description Module, enabling the mobile application to provide visually impaired users with information about the distance of detected objects from the camera. By leveraging the depth maps generated by MIDAS and the segmentation masks obtained through instant segmentation using YOLOv8, the application can estimate the approximate distances to the objects within the captured scene.

The triangulation process begins by converting the segmented regions of the depth map into grayscale heat maps. Each heat map represents the relative depth and intensity of a specific object detected within the scene. To obtain a more accurate distance estimation, the maximum grayscale value within each object's segmented region is extracted. This maximum grayscale value serves as a proxy for the object's depth.

With the maximum grayscale value obtained, the triangulation formula is applied to estimate the approximate distance of the object from the camera. The formula takes into account the known dimensions of the camera sensor and the angle subtended by the object in the image. By leveraging the principles of similar triangles, the formula calculates the distance to the object.

The distance to an object can be estimated using the following formula:
distance to object = (object width in real life * focal length) / (object pixel width in image * depth value).

To determine the object width in real life, we created a Python dictionary based on the average widths of objects in the COCO dataset. Each object in the dictionary is associated with its corresponding average width.

The focal length is a constant value specific to the camera used for capturing the images. We will elaborate on the method for calculating the focal length later in the explanation.
The object pixel width in the image is determined by measuring the width of the object using its bounding box in the segmented image.

The depth value is obtained from the segmented heat map grayscale image generated by the Midas model.

To handle variations in object orientation, we consider the maximum width and height values for both the object width in real life and the object pixel width in the image. This accounts for situations where the object may be placed vertically or horizontally in the image.

Initially, we attempted to use the constant focal length value provided by the camera in the mobile configuration. However, we found that the results were not realistic. To address this, we decided to calculate the focal length value by isolating it on one side of the equation. We conducted experiments using objects with known distances in real life (measured in centimeters) and placed them at specific distances.

By plotting a graph of the trials, we obtained a linear equation of the form Y = a * X + b, where Y represents the distance to the object (in centimeters), X represents the calculated value (object width in real life) / (object pixel width in image * depth value), and "a" and "b" are constants.

Our aim was to determine the values of "a" and "b" in this linear equation. Although the equation does not have to be linear, when we plotted the graph of the trials, we observed that the resulting output followed a linear pattern. Therefore, we obtained the values of "a" and "b" to establish a relationship between the calculated value and the actual distance to the object.

Table 4-2: Distance estimation for real life objects

Name	Real Distance	Width	Height	Object Real (Avg) Width	Depth Value	X	Prediction
Apple	38	166	202	10	0.062745098	0.788985149	74.2
Chair	100	851	1126	100	0.101960784	1.742041262	123.8
Chair	135	466	731	100	0.090196078	1.516683519	114.2
Chair	151	598	841	100	0.08627451	1.378229381	107.6
Chair	170	406	696	100	0.050980392	2.818302387	151.6
Chair	225	378	506	100	0.058823529	3.359683794	154.3
Chair	50	999	1526	100	0.105882353	0.618901995	62.9
Chair	84	1188	1156	100	0.121568627	0.5286508	56.6
Chair	126	554	888	100	0.090196078	1.24853114	101
Chair	144	461	757	100	0.078431373	1.684280053	121.5
Cup	100	109	103	11	0.047058824	2.144495413	137.7
Cup	150	85	77	11	0.031372549	4.125	145.2
Cup	25	425	395	11	0.141176471	0.183333333	30.5
Cup	50	215	199	11	0.121568627	0.420855214	48.8
Fork	45	77	273	30	0.078431373	1.401098901	108.7
Laptop	40	1198	1064	40	1	0.033388982	18.3
Laptop	46	823	646	40	0.109803922	0.442631488	50.4
Laptop	88	576	526	40	0.08627451	0.804924242	75.2
Laptop	122	323	306	40	0.082352941	1.503759398	113.6
Mouse	22	373	283	10	0.133333333	0.201072386	32
Mouse	32	216	297	10	0.117647059	0.286195286	38.6
Mouse	47	284	114	10	0.078431373	0.514112903	55.6
Refrigerator	81	978	1568	200	0.137254902	0.929300292	83
Refrigerator	108	816	1466	200	0.133333333	1.02319236	88.6
Refrigerator	135	718	1381	200	0.11372549	1.273440036	102.3
Spoon	35	127	511	30	0.08627451	0.6804839	67.1
Suitcase	60	445	563	50	0.105882353	0.838760608	77.4
Suitcase	70	469	719	50	0.11372549	0.611481464	62.4
Suitcase	115	469	373	50	0.098039216	1.087420043	92.3
Suitcase	130	277	309	50	0.066666667	2.427184466	144.9
Suitcase	146	338	336	50	0.066666667	2.218934911	139.8
Vase	72	191	298	80	0.094117647	2.852348993	152
Vase	105	136	226	80	0.070588235	5.014749263	115.6

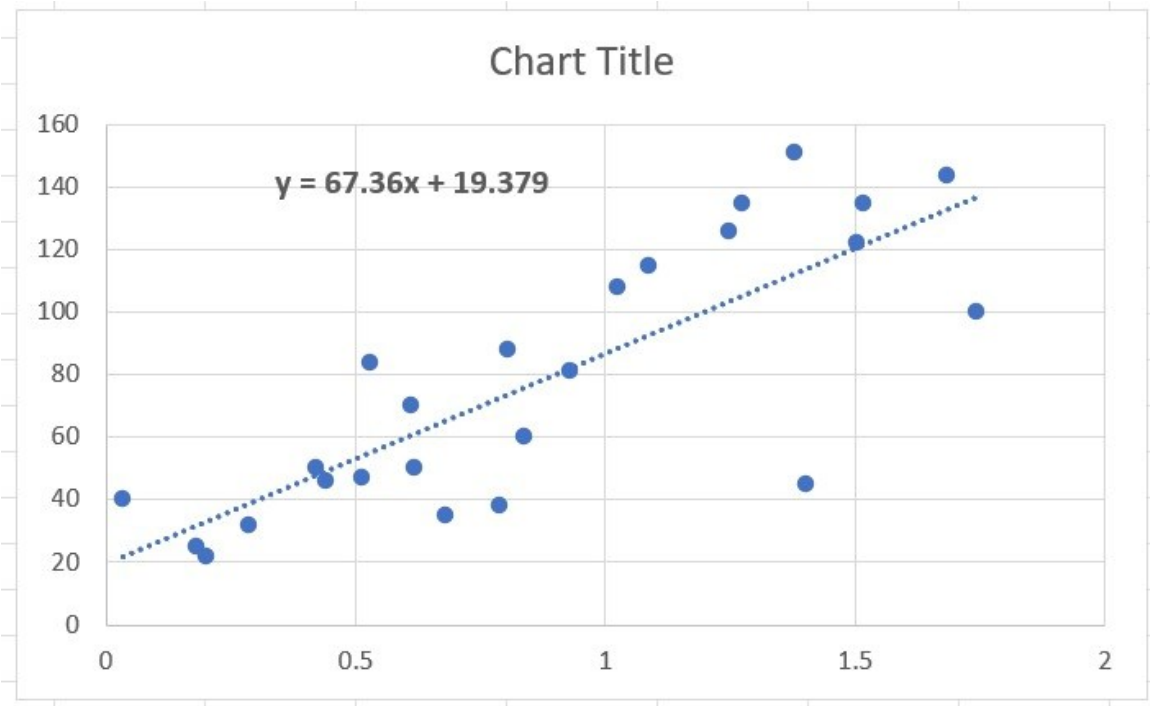


Figure 4-8: Plotting Triangularization equation

The value of “a” is 67.36 and the value of “b” is 19.379.
The final equation is Y = 67.36*X + 19.379

The estimated distance is then compared to a predefined threshold, typically set at 1 meter in this scenario. If the estimated distance is below the threshold, the application informs the user that the object is closer than 1 meter. Similarly, if the estimated distance exceeds the threshold, the application indicates that the object is further than 1 meter away. This information is relayed to the user, allowing them to comprehend the spatial layout of the detected objects and make informed decisions about their surroundings.

4.3.3 Design Constraints

First, COCO dataset does not contain all the categories that a visually impaired will face in his everyday life. So, users will face some situations where the module will not be fully helpful.

In addition, it is important to note that the triangulation and distance estimation process provides an approximation of the distance to the objects. Several factors can affect the accuracy of the estimation, including the precision of the depth maps obtained from MIDAS, the quality of the segmentation masks, and the assumptions made in the triangulation formula. Additionally, environmental conditions and camera calibration can

introduce errors in distance estimation. Nevertheless, the distance estimation capability enhances the scene description module's functionality, providing visually impaired users with valuable information about object proximity and spatial awareness.

4.4 Clothes Descriptor

The Clothes Description Module is an integral part of the mobile application designed to assist visually impaired individuals in understanding and interacting with clothing items. This module comprises four stages: dataset preparation, Mask R-CNN, texture detection, and color detection.

4.4.1 Functional Description

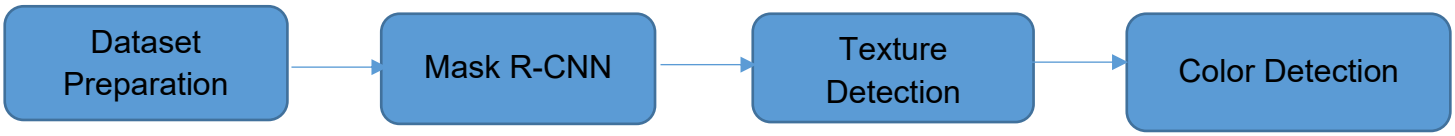


Figure 4-9 Clothes Descriptor Components

In the dataset preparation stage, multiple datasets, including DeepFashion2 and a custom clothing detection dataset, are utilized to gather a diverse collection of clothing images with corresponding annotations.

The Mask R-CNN stage involves training the model on a subset of the dataset, where images are processed to extract clothing segments with their respective bounding boxes and segmentation masks.

In the texture detection stage, traditional machine learning techniques are employed, including feature extraction with GLCM, HOG, and DAISY, with a focus on GLCM and DAISY features. The extracted features undergo concatenation, dimensionality reduction using PCA, and standardization.

Model selection and training are conducted using various algorithms such as KNN, ANN, SVM, HMM, Random Forest, Adaboost, and XGBoost, enabling the identification and classification of different clothing textures.

The final stage, color detection, involves analyzing the segmented clothing images to identify dominant colors and determine their proximity to predefined RGB values, enabling accurate color identification.

Through these stages, the Clothes Description Module empowers visually impaired users to explore and comprehend clothing items, providing valuable information about their texture and color.

4.4.2 Modular Decomposition

4.4.2.1 Dataset Preparation

1. DeepFashion2 Dataset

The DeepFashion2 dataset is a widely recognized dataset in the field of fashion analysis and understanding. It contains a large collection of images with associated annotations, including bounding boxes, segmentation masks, and attribute labels for various fashion items. This dataset provides a diverse range of clothing categories, styles, and poses, making it valuable for training clothes description models.

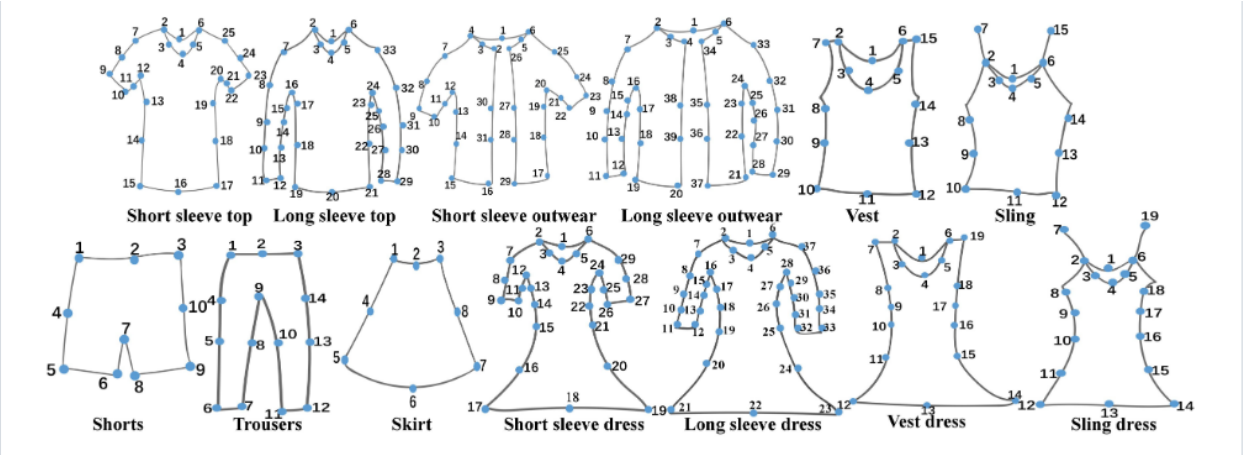


Figure 4-10: Definitions of landmarks and skeletons

	Train	Validation	Test	Overall
images	390,884	33,669	67,342	491,895
bboxes	636,624	54,910	109,198	800,732
landmarks	636,624	54,910	109,198	800,732
masks	636,624	54,910	109,198	800,732
pairs	685,584	query: 12,550 gallery: 37183	query: 24,402 gallery: 75,347	873,234

Figure 4-11: Statistics of Deepfashion2 (1)

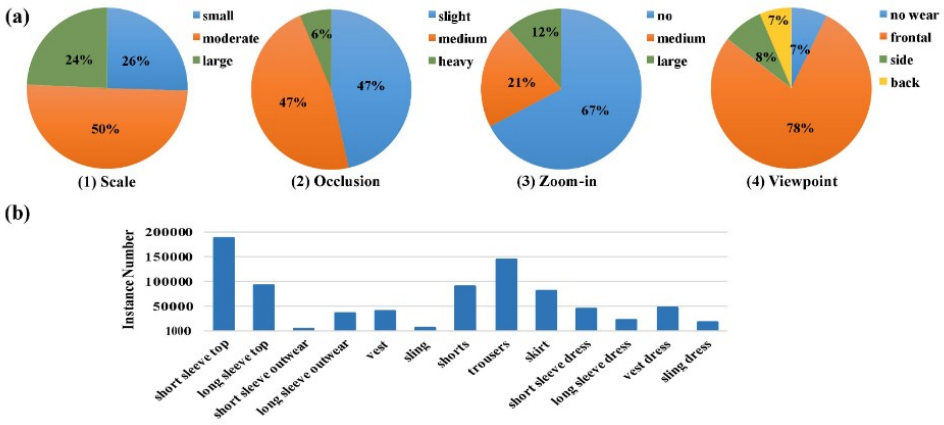


Figure 4-12:Statistics of Deepfashion2 (2)

2. Clothing Detection E-commerce Dataset

The Clothing Detection E-commerce dataset is a curated dataset specifically designed for clothing detection and analysis in e-commerce settings. It consists of many images extracted from e-commerce websites, focusing on clothing products. The dataset includes images with corresponding bounding box annotations, which indicate the locations of clothing items within the images. This dataset supplements the DeepFashion2 dataset by providing additional images and variations in clothing styles and contexts.

3. Custom Texture Dataset

Table 4-3: Clothing Detection E-commerce Dataset Classes

trousers	coat	jacket	belt	necklace
vest	ring	shirt	hat	bracelet
tie	bag	watch	scarf	boot
sportshoes	sunglasses	earrings	skirt	dress
socks and tight	shoes	underwear	backpack	short
night morning	gloves and mitten	wallet and purse	cufflinks	swimwear
makeup	suitcase	jumpsuit	suspenders	pouchbag

To incorporate texture analysis into the clothes description module, a custom dataset was created. This dataset contains images of different textures commonly found in clothing materials. The dataset was created by using an image download package, which facilitated the collection of images representing various textures such as cotton, silk, denim, wool, and leather. These textures were selected based on their relevance to clothing classification and their importance in describing clothing characteristics.

The combination of these three datasets provides a comprehensive collection of images and annotations necessary for training and evaluating the clothes description models. By utilizing these datasets, we can leverage a wide variety of clothing styles, attributes, and textures, enabling accurate and detailed descriptions of clothing items within the mobile application.

4.4.2.2 Masr R-CNN

In this stage, the Mask R-CNN model was trained on a subset of the available dataset due to resource limitations. A total of 20,000 images from the Clothing Detection Ecommerce dataset and 30,000 images from the DeepFashion dataset were used for training. Prior to training, the dataset was prepared to ensure that the annotations were in the required format for Mask R-CNN. Each image was accompanied by a corresponding JSON file containing the segmentation information for each object, the category of the object, and the coordinates of the bounding box.

The training process involved optimizing the parameters of the Mask R-CNN model through iterative epochs. The model was trained for 100 epochs, where each epoch represents a complete iteration through the entire dataset. During training, the model gradually learned to recognize and segment different clothing items based on the provided annotations. The training process aimed to improve the model's accuracy and ability to accurately classify and segment clothing objects in unseen images.

After training, the trained Mask R-CNN model was saved as an .h5 file, which preserves the learned weights and architecture of the model. This saved model could later be loaded and utilized for clothing description tasks in the mobile application.

Mask R-CNN, short for Mask Region Convolutional Neural Network, is a popular deep learning architecture for object detection and instance segmentation tasks. It extends the Faster R-CNN framework by incorporating an additional branch that generates pixel-level segmentation masks for each detected object. This allows for precise delineation of object boundaries and accurate identification of object instances within an image.
The Mask R-CNN architecture consists of two main components: the Region Proposal Network (RPN) and the Mask Head. The RPN is responsible for generating region proposals, which are potential bounding box regions that might contain objects.

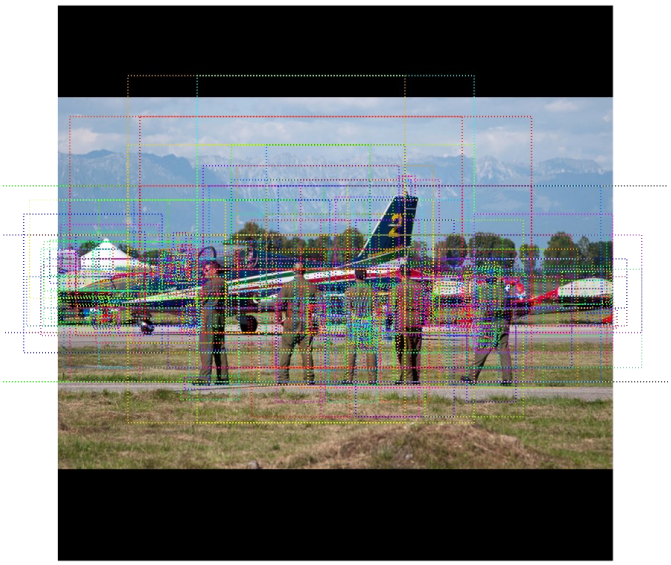


Figure 4-13: Mask R-CNN RPN

The Mask Head, on the other hand, takes these region proposals and performs classification, bounding box refinement, and pixel-wise segmentation tasks.

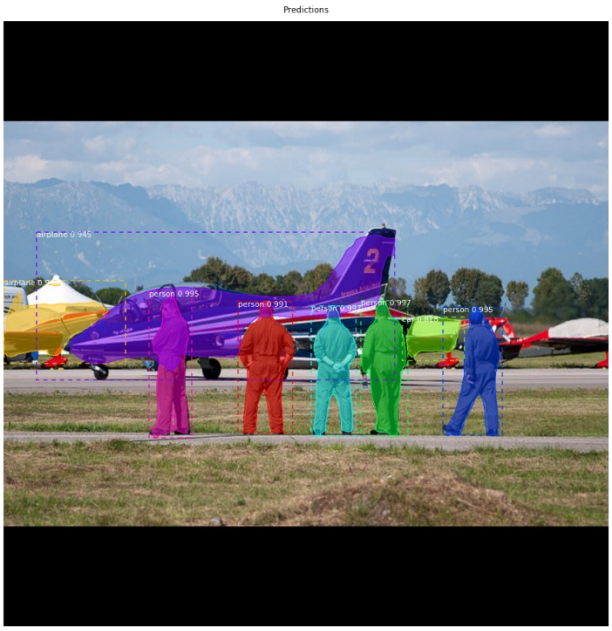


Figure 4-14: Mask R-CNN Head

During inference, the trained Mask R-CNN model takes an input image and passes it through the network. The RPN proposes regions of interest, which are then refined by the bounding box regression layer. Simultaneously, the Mask Head generates a binary mask for each proposed region, indicating the presence or absence of an object at each pixel location. This results in both object detection and pixel-level segmentation of the image. The training process involves optimizing the model's parameters by comparing the predicted masks and bounding boxes with the ground truth annotations. This is done through a combination of classification loss, bounding box regression loss, and segmentation mask loss. The model is trained using gradient descent-based optimization techniques, such as stochastic gradient descent (SGD) or Adam optimizer, to minimize the overall loss.

By training Mask R-CNN on the annotated clothing dataset, the model can learn to accurately detect and segment various clothing items, enabling the subsequent stages of texture detection and color detection in the clothes Description module of the mobile application.

Upon completing this stage, the trained Mask R-CNN model can process an input image and extract the detected clothing items along with their respective bounding boxes and segmentation masks. This enables the extraction of individual segments for each recognized clothing object. These segmented regions can then be seamlessly passed on to the subsequent stages of texture detection and color detection, where further analysis

and classification of the clothing segments take place. By leveraging the power of Mask R-CNN, the mobile application can accurately isolate and process each clothing item.

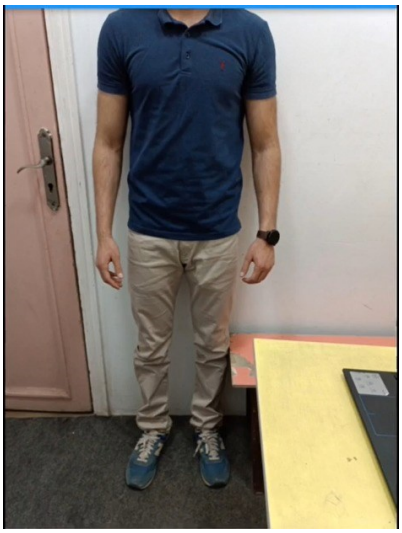
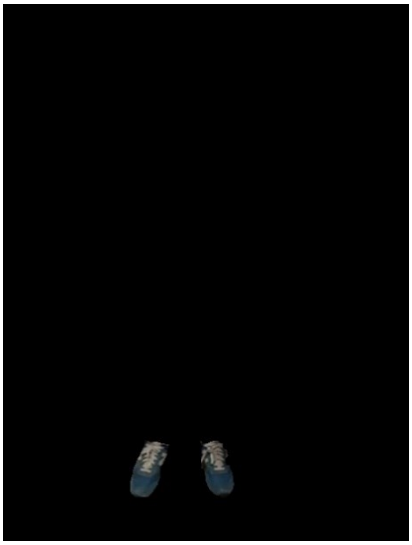


Figure 4-15 Input Clothes Image

Figure 4-16 Mask R-CNN segmentation result



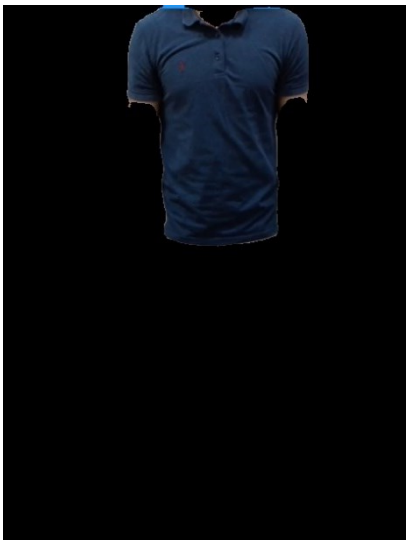


Figure 4-17 Clothes Segments

4.4.2.3 Texture Detection

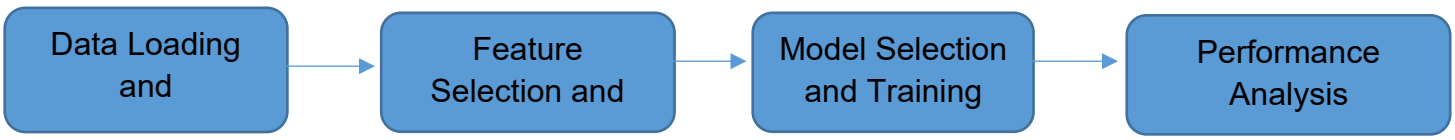


Figure 4-18 Texture Detection Pipeline

1. Data Loading and preprocessing

In the texture detection stage, the first step is to load and preprocess the custom texture dataset that was previously mentioned. This dataset consists of images representing different textures. The following steps are performed during data loading and preprocessing:

a) Loading the Dataset: The custom texture dataset is loaded into memory. This dataset contains a collection of images, each representing a specific texture category.

b) Image Resizing: To ensure efficient processing and to mitigate the impact of high-dimensional feature spaces, the images in the dataset are resized while preserving the aspect ratio. Resizing the images helps reduce the number of features and prevents overfitting during model training. It also contributes to speeding up the training process.

c) Dataset Split: The loaded dataset is split into three distinct subsets:

Training Set: This subset is used to train the texture detection model. It contains a majority of the images from the dataset and serves as the basis for learning the patterns and characteristics of different textures.

Validation Set: The validation set is utilized for tuning the hyperparameters of the texture detection model. It helps in optimizing the model's performance by selecting the most effective set of hyperparameters that generalize well to unseen data.

Test Set: The test set is reserved for evaluating the performance of the trained models. It consists of a separate collection of images that were not used during training or validation. The test set provides an unbiased assessment of the model's ability to accurately detect and classify textures.

d) Shuffling the Training Set: To prevent any potential bias during model training, the training set is shuffled randomly. This ensures that the images within the training set are not ordered by class, which could potentially influence the learning process. Shuffling the training set helps the model learn the distinguishing features of different textures without any systematic ordering.

By performing data loading and preprocessing, the custom texture dataset is prepared for the subsequent stages of feature selection and extraction, model selection and training, and performance analysis. This initial stage establishes a solid foundation for the texture detection pipeline, enabling the subsequent stages to be executed effectively.

2. Feature Extraction and Selection

In the texture detection stage, feature extraction and selection play a crucial role in capturing the distinctive characteristics of different textures. In this stage, several feature extraction techniques were explored, including GLCM, HOG, and DAISY. The following is an overview of each of these techniques, with a specific focus on GLCM and DAISY due to their favorable performance and their combined usage after performing PCA:

a) GLCM (Gray-Level Co-occurrence Matrix):

GLCM is a widely used technique in texture analysis that captures the statistical relationship between pairs of pixels in an image. It quantifies the frequency of pixel intensity combinations, providing information about texture properties such as contrast, homogeneity, energy, and entropy. The GLCM is constructed by computing the occurrence of different pixel intensity pairs within a specified distance and angle in the image.

The GLCM matrix is typically symmetric, as it considers the co-occurrence of pixel pairs in both the horizontal and vertical directions. Each element of the GLCM represents the number of times a specific pixel intensity pair occurs at a given distance and angle. From the GLCM, various texture features can be derived to describe different aspects of the texture.

Some of the commonly extracted texture features from GLCM include:
Contrast: Measures the intensity contrast between neighboring pixels in the image. It is calculated as the sum of squared differences between pixel intensity values within the GLCM.

Energy (Angular Second Moment): Represents the uniformity or smoothness of the texture. It is calculated as the sum of squared GLCM elements and signifies the local homogeneity.
Homogeneity (Inverse Difference Moment): Reflects the similarity or closeness of pixel intensity values. It is calculated as the sum of inverse differences between pixel intensity values within the GLCM.

Entropy: Measures the amount of information or randomness in the texture. It is calculated by summing the negative values of each GLCM element multiplied by their logarithm.

b) DAISY (Descriptor of Accurate and Intuitive Shape Yield):

DAISY is a local image descriptor that captures both spatial and gradient information. It computes dense descriptors at multiple scales and orientations, providing a representation that is robust to geometric and photometric variations. DAISY features are particularly effective in capturing texture patterns, edges, and corners. By extracting DAISY features, the texture detection model can effectively differentiate textures based on their distinctive local structures. The number of extracted daisy features was 272 per image. That's why we decided to perform PCA.

c) PCA (Principal Component Analysis):

The combination of GLCM and DAISY features after performing Principal Component Analysis (PCA) allows for the fusion of their complementary information. PCA is a dimensionality reduction technique that projects the original feature space onto a lower-dimensional space while preserving the most informative aspects of the data. This step helps to reduce the dimensionality of the feature space and improve the efficiency of subsequent processing. After performing PCA on daisy features, the number of features was reduced to 15. So, we have a total of 20 features per image after concatenating GLCM with PCA daisy features.

D) Standardization:

Following the feature extraction and selection, the extracted GLCM and DAISY features are concatenated into a unified feature vector. To ensure that all features contribute equally during model training, standardization is performed. Standardization scales the features to have zero mean and unit variance, ensuring that they are on the same scale and have similar weightage in the subsequent stages.

By incorporating GLCM and DAISY features, along with PCA and standardization, the texture detection model benefits from their combined discriminative power, capturing both statistical and local structural information. These enhanced features serve as a vital input for the subsequent stages of model selection and training, leading to improved texture classification performance.

3. Model Selection and Training

In the model selection and training stage, we explored several machine learning algorithms to identify the best approach for texture detection. The following models were considered and trained on the preprocessed dataset:

K-Nearest Neighbors (KNN): KNN is a non-parametric algorithm that classifies an input sample based on the majority vote of its nearest neighbors. The number of neighbors (K) is a hyperparameter that affects the model's performance.

Artificial Neural Network (ANN): We employed an ANN with two hidden layers for texture detection. ANNs are versatile models inspired by the structure of the human brain. They consist of interconnected nodes (neurons) organized in layers and learn to recognize patterns through iterative training.

Support Vector Machines (SVM): SVM is a supervised learning algorithm that constructs hyperplanes to separate different classes in the feature space. It maximizes the margin between classes, aiming to achieve optimal separation.

Hidden Markov Model (HMM): HMM is a probabilistic model widely used in sequence analysis. It models the temporal dependencies of data, making it suitable for tasks involving sequential patterns, such as speech recognition and gesture recognition.

Random Forest: Random Forest is an ensemble learning method that constructs multiple decision trees and combines their predictions to make the final classification. It reduces overfitting and improves generalization by averaging the results of individual trees.

Adaboost: Adaboost is an ensemble technique that combines weak classifiers to form a strong classifier. It assigns higher weights to misclassified samples in each iteration, focusing on difficult examples to improve overall performance.

XGBoost: XGBoost is an optimized implementation of gradient boosting, which is an ensemble learning technique. It builds an additive model by sequentially adding weak learners and minimizing a loss function.

For each of these models, we conducted the training process using the preprocessed dataset and the selected features. The training involved optimizing the respective model-specific hyperparameters and adjusting the parameters to achieve the best performance.

4. Performance Analysis

In the performance analysis stage, we evaluated the performance of the trained texture detection models using various metrics. The metrics we utilized include:

Micro Average Precision: It calculates the precision by considering the total true positives, false positives, and false negatives across all classes. It provides an overall measure of precision for the entire dataset.

Micro Average Recall: This metric calculates the recall by considering the total true positives, false positives, and false negatives across all classes. It gives an overall measure of recall for the entire dataset.

Micro Average F1-Score: The F1-Score is the harmonic means of precision and recall. The micro-average F1-Score considers the total true positives, false positives, and false negatives across all classes, providing an overall measure of the model's effectiveness.

Macro Average Precision: It calculates the precision for each class individually and then takes the average across all classes. This metric gives an equal weight to each class, regardless of its size or frequency.

Macro Average Recall: Like macro average precision, this metric calculates the recall for each class individually and then takes the average across all classes. It provides an equal weight to each class in terms of recall.

Macro Average F1-Score: The macro-average F1-Score calculates the F1-Score for each class individually and then takes the average across all classes. It provides an equal weight to each class in terms of F1-Score.

Weighted Macro Average Precision: This metric calculates the precision for each class individually, considering the class's frequency, and then takes the weighted average across all classes. It gives more weight to classes with larger sample sizes.

Weighted Macro Average Recall: Like weighted macro average precision, this metric calculates the recall for each class individually, considering the class's frequency, and then takes the weighted average across all classes.

Accuracy: Accuracy measures the proportion of correctly classified samples to the total number of samples. It provides an overall measure of how well the model performs in terms of correctly predicting the texture classes.

When predicting the texture of different cloth segments, the segmented image undergoes a series of preprocessing steps. Firstly, the image is resized to ensure uniformity in the input data.

Next, feature extraction is performed using GLCM and DAISY, capturing statistical properties and local image structures respectively. These extracted features are then subjected to PCA for dimensionality reduction. Subsequently, the features are normalized using the mean and standard deviation from the training set. Finally, the preprocessed features are fed into the trained model to predict the texture category of the cloth segment, which can be cotton, denim, wool, silk, or leather. This process empowers the mobile application to accurately identify and provide information about the texture of various cloth segments to visually impaired users.

4.4.2.4 Color Detection

The final stage in the pipeline of the clothes description module is color detection. This stage focuses on determining the color of the cloth segment.

To begin, the segmented image is processed to identify the dominant colors present within the cloth segment. The count of unique dominant colors is calculated, providing insights into the color diversity of the cloth.

Next, the top three dominant colors are selected for further analysis. These colors represent the most prominent hues within the cloth segment.

To determine the actual color of the cloth, a comparison is made between the selected dominant colors and the RGB values of the ten most principal colors. The principal colors serve as reference points for common color categories such as red, blue, green, yellow, etc. By measuring the proximity of the dominant colors to the RGB values of the principal colors, the closest match is identified as the color of the cloth segment.

This color detection process enhances the mobile application's ability to accurately describe the color of various cloth segments to visually impaired users.

4.4.3 Design Constraints

The design constraint of the Clothes Description Module is that the accuracy of the clothes detection and segmentation models is limited due to training on a subset of the dataset. This constraint may result in potential missed detections or inaccuracies in segmenting clothing items.

Furthermore, the texture module may not perform as expected, as it requires a close examination of the texture and was trained specifically for such scenarios. In cases where the clothing has multiple colors, the module may detect the three most dominant colors present in the cloth, providing insights into its color composition.

These design constraints highlight the need for further improvements and refinements in the training process and model development to enhance the accuracy and performance of the clothes detection, segmentation, and texture analysis within the module.

4.5 Text Reader

The text recognition module uses deep learning to extract text from images, enhancing accessibility for visually impaired individuals. Integrated into assistive technologies like screen readers and apps, it recognizes various languages and styles, promoting inclusivity and improved quality of life.

4.5.1 Functional Description

The text recognition module uses a 3-layered CNN trained on the EMNIST dataset to extract text from images for OCR, document digitization, and assistive technologies. The module converts image-based text into machine-readable formats.

The diverse EMNIST dataset, an extension of MNIST, contains handwritten characters from multiple languages, ensuring the module can recognize various characters and text styles.

The 3-layered CNN architecture consists of the following layers:

1. Convolutional Layer: Detects local features using convolutional filters and ReLU activation, improving learning efficiency.

2. Pooling Layer: Reduces spatial dimensions with max-pooling, lowering computational complexity and enabling position-invariant recognition.

3. Fully Connected Layer: Acts as a classifier, using SoftMax activation to produce probability scores for character classes.

The 3-layered CNN-based text recognition module efficiently extracts text from images, providing a robust solution for various applications.

4.5.2 Modular Decomposition

This section provides an in-depth discussion of the three main stages of the text recognition module: preprocessing the data, creating and training the CNN model, and creating bounding boxes for letter detection in real documents.

4.5.2.1 Stage 1: Preprocessing and Data Augmentation

The first stage of the text recognition module involves several steps to preprocess the EMNIST dataset and prepare it for training the 3-layered CNN architecture. The following steps are carried out in this stage:

1. Removing N/A: The EMNIST dataset may contain instances where character labels are missing or marked as not applicable (N/A). These instances can negatively impact the model's training as they do not provide meaningful information for learning. To ensure a clean and accurate dataset, all instances with N/A labels are removed during the preprocessing stage.

```
['N/A', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']
27 classes
torch.Size([124800, 28, 28])
torch.Size([124800, 1, 28, 28])
tensor(0)

tensor([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
        19, 20, 21, 22, 23, 24, 25, 26])
```

Figure 4-19 the N/A that is found in the EMNIST dataset making them 26 classes.

2. Normalization: The pixel values in the images are normalized to a range of 0 to 1 by dividing each pixel value by 255.

3. Data Augmentation: The model uses rotation and lighting adjustments to improve robustness and adapt to different orientations and lighting conditions.

4. Reshaping and Grayscale Conversion: Images are converted to grayscale and reshaped to fit the CNN's expected input dimensions.

5. Dataset Splitting: The preprocessed EMNIST dataset is divided into training, validation, and testing subsets using stratified sampling for balanced character distributions.

4.5.2.2 Stage 2: Creating and Training the CNN Model

The second stage of the text recognition module involves creating and training a CNN model for character recognition. The provided code defines a custom neural network called EMNISTNet that uses the PyTorch framework. Let's break down the code and discuss each layer and the activation functions used.

EMNISTNet CNN Architecture

The EMNISTNet architecture consists of three convolutional layers, three batch normalization layers, three fully connected layers, and two dropout layers.

1. Convolutional1. Convolutional Layers: Three layers (conv1, conv2, conv3) with increasing output channels, detecting low to complex features.
2. Batch Normalization Layers: bnorm1, bnorm2, bnorm3 normalize activations, improving training and generalization.
3. Fully Connected Layers: Three layers (fc1, fc2, fc3) refine abstract representations and classify input images into 26 letter classes.
4. Dropout Layers: dropout1 and dropout2 prevent overfitting by randomly dropping out units with a rate of 0.5.

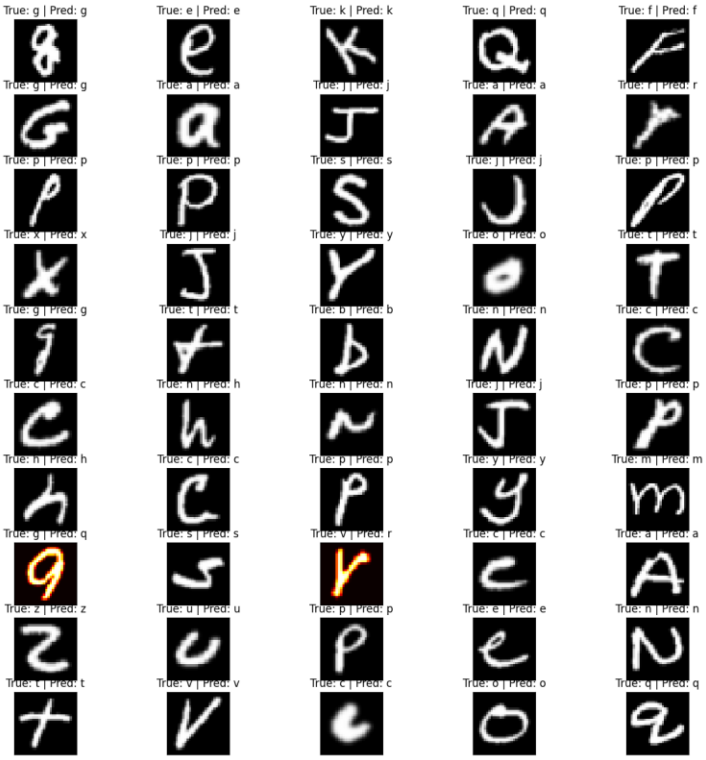


Figure 4-20 The result of the trained module it can recognize most of the letters

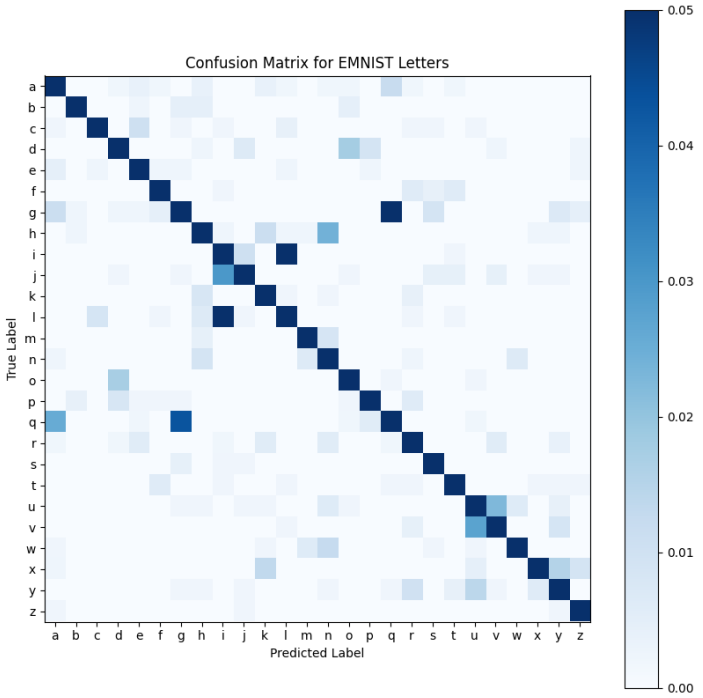


Figure 4-21 The resulting grid representing the performance of the OCR model on the EMNIST Letters dataset.

4.5.2.3 Stage 3: Post-processing and Text Output

The last stage of the text recognition module involves post-processing the images and predicting the characters using the trained CNN model. The provided code includes several functions that work together to achieve this goal:

1. preprocess document image: Preprocesses the image by converting it to grayscale and applying binary inversion and thresholding using Otsu's method. The result is a binary image where the text is white, and the background is black.

2. add padding: This function takes a binary image and an optional padding value (default is 2) and adds padding around the image. This is useful for ensuring that the characters do not touch the borders of the image, which can help improve character recognition.

3. segment lines and segment characters: These functions segment the preprocessed image into lines and characters, respectively.

4. skeletonize: This function takes a binary image and skeletonizes the text, which can help improve character recognition by simplifying the text's structure and reducing the impact of noise.

5. resize character: This function takes a character image, resizes it to a specified size (default is 28x28), and adds padding around the image. This is necessary because the CNN model expects input images to have a size of 28x28.

6. recognize character: predicts a character using a trained CNN model by preprocessing the input character image, converting it to a PyTorch tensor, and identifying the highest probability class among the 26 letter classes.

7.extract text from document image: is the main function that combines preprocessing, segmentation, and character recognition using a trained CNN model to output recognized text from a document image. It returns the output text and the last recognized character as a tuple.

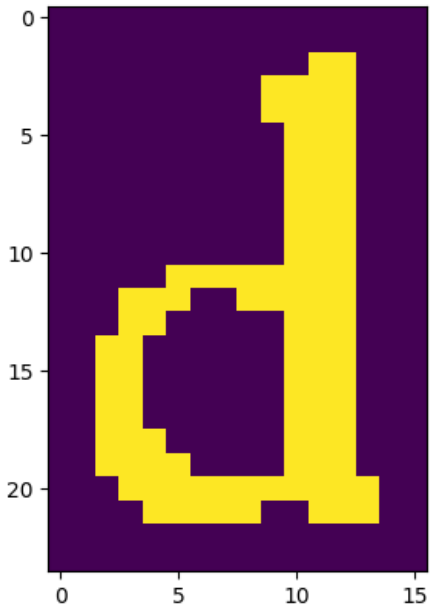


Figure 4-22 Separated letter D

Extract text from document image first preprocesses the image and then segments the lines. For each line, it segments the characters using and then recognizes each character. The recognized characters are concatenated to form the output text, with line breaks added between the lines of text.

4.5.3 Design Constraints

While the described approach for text recognition using a CNN model on the EMNIST Letters dataset is effective, there are some constraints and limitations:

1. Limited character set: The model is trained on the EMNIST Letters dataset, which only includes uppercase English letters (26 classes). It lacks support for lowercase letters, digits, punctuation, and special characters.

2. Sensitive to image preprocessing: The accuracy of the text recognition depends heavily on the quality of the image preprocessing and segmentation steps.

3. Inability to handle distorted or noisy text: The model may struggle with recognizing text that is distorted, noisy, or written in unusual fonts, as the EMNIST dataset mostly contains clean and normalized character images.

4. Lack of context: The approach recognizes individual characters independently without considering the context of the surrounding characters.

5. No support for different languages: This approach is designed for English text recognition. To support other languages, the model would need to be trained on a dataset that includes characters and writing systems from those languages.

6. Computational complexity: The CNN model and the image processing steps can be computationally intensive, especially for large images or real-time applications.

7. Lack of rotation and scale invariance: The model may not perform well when dealing with text that is rotated or scaled differently from the training data.

4.6 Currency Recognizer

The currency recognition model, which utilizes KNN with histogram, texture, and ORB features, plays a significant role in improving the quality of life for visually impaired individuals.

By providing a reliable and accessible solution, the currency recognition model empowers visually impaired users with greater independence and confidence in their daily financial transactions. Here are some key benefits of this model for visually impaired individuals:

1. Enhanced Autonomy: The model enables visually impaired users to manage their finances independently, without relying on others for assistance.

2. Reduced Risk of Fraud: The model can help protect visually impaired individuals from potential fraud or exploitation by accurately identifying currency denominations.

3. Increased Confidence: Being able to handle financial transactions effectively boosts the confidence of visually impaired people.

4. Ease of Use: The currency recognition model can be integrated into user-friendly applications or devices, making it easily accessible for visually impaired users.

5. Inclusion and Accessibility: By providing an effective currency recognition solution, the model promotes financial inclusion and accessibility for visually impaired people.

4.6.1 Functional Description

The currency recognition model for visually impaired people is designed to identify different currency denominations accurately and efficiently. Here's a step-by-step functional description of the model:

1. Preprocessing: The model applies techniques like resizing, denoising, and normalization to enhance input images and ensure consistency, improving overall performance.

2. Feature Extraction: The model extracts histogram, texture, and ORB features to capture color distribution, patterns, and invariant information, respectively.

3. KNN Classification: Once the feature vectors are obtained, they are used as input for the k-Nearest Neighbors (KNN) algorithm.

4.6.2 Modular Decomposition

4.6.2.1 First Stage: Data Preprocessing

In the currency recognition model, the first stage focuses on data preprocessing. This stage plays a critical role in preparing the input dataset and ensuring that the images are in a suitable format for subsequent feature extraction. The provided code contains several image manipulation techniques intended to augment the dataset and enhance the model's robustness. The following is a detailed overview of the code:

1. Rotation and Scaling: The functions rotate (img, angle) and scale (img, scale) are responsible for applying rotation and scaling transformations to the input images, respectively.

2. Noise Addition: The noise (noise_typ, image) function introduces various types of noise to the input image based on the specified noise_typ. Supported noise types include Gaussian noise, salt and pepper noise, Poisson noise, and speckle noise.

3. Blurring: The blur (img, blur_type) function applies different blurring techniques to the input image according to the blur_type parameter. The available blur types comprise average blur, Gaussian blur, median blur, and bilateral blur.

4. Lighting Adjustment: The lighting (img, lighting type) function modifies the lighting conditions of the input image based on the lighting type parameter. The lighting adjustments encompass brightness, contrast, gamma correction, histogram equalization.

4.6.2.2 Second Stage: Feature Extraction

The second stage of the currency recognition model for visually impaired people involves extracting features from the preprocessed images. The feature extraction process comprises three steps:

1. Histogram Features: The model extracts color information using histograms for each color channel, providing insight into color composition and intensity patterns crucial for currency recognition.

2. Texture Features: The model captures structural and textural details using methods like GLCM or LBP, analyzing spatial relationships between pixels and encoding patterns essential for distinguishing currency denominations.

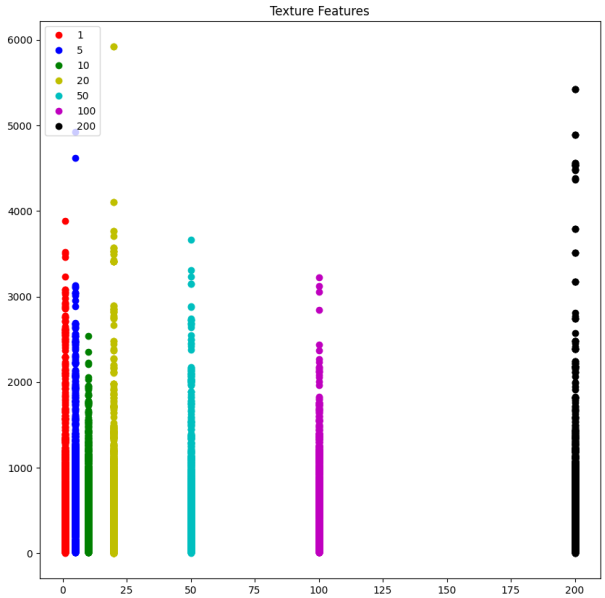


Figure 4-23 GLCM of each currency which shows difference in texture for each currency.

3. ORB Features: The Oriented FAST and Rotated BRIEF (ORB) algorithm is employed to extract key point and descriptor information from the currency images. The ORB method is a fast and efficient feature detector and descriptor extractor that is invariant to rotation, scale, and illumination changes. It identifies distinctive points (key points) in the images and computes a binary descriptor for each key point, capturing the local patterns

around them. These key points and their corresponding descriptors serve as an essential input for the classification stage.

After extracting the histogram, texture, and ORB features, the model combines them to create a comprehensive feature vector for each currency image. This feature vector is then utilized as input for the k-Nearest Neighbors (KNN) algorithm during the classification stage. The second stage of feature extraction is vital as it enables the model to capture various visual attributes of the banknotes, ultimately leading to accurate currency recognition and improved assistance for visually impaired users.

4.6.2.3 Third Stage: Dimensionality Reduction using PCA.

After the feature extraction stage, the currency recognition model proceeds to the third stage, which involves dimensionality reduction using Principal Component Analysis (PCA). The high-dimensional feature vector obtained from the previous stage may contain redundant information and increase computational complexity. PCA helps to address these issues by transforming the original feature space into a lower-dimensional space while retaining most of the information.

1. Standardize the Feature Data: Before applying PCA, it is important to standardize the feature data to ensure that each feature contributes equally to the analysis. Standardization scales the feature values to have a mean of 0 and a standard deviation of 1.

2. Determine the Optimal Number of Principal Components: To find the best number of principal components, calculate the explained variance ratios for each component and plot the cumulative explained variance against the number of components. From the plot, you can visually inspect the point at which the cumulative explained variance reaches your desired threshold (e.g., 95%). The `n_components` variable will contain the best number of components based on the threshold.

3. Apply PCA with Optimal Number of Components: Perform PCA using the determined number of components (`n_components`). This step transforms the high-dimensional feature vector into a lower-dimensional representation, where the principal components capture the most important variations in the data.

4. Update the Feature Vector: Replace the original high-dimensional feature vector with the lower-dimensional representation obtained from PCA. This compact representation retains the most relevant information while reducing computational complexity and mitigating the risk of overfitting.

In this third stage, PCA effectively reduces the dimensionality of the feature data, leading to a more efficient and robust currency recognition model. The lower-dimensional representation serves as input for the k-Nearest Neighbors (KNN) algorithm during the classification stage, enabling accurate currency recognition and enhanced assistance for visually impaired users.

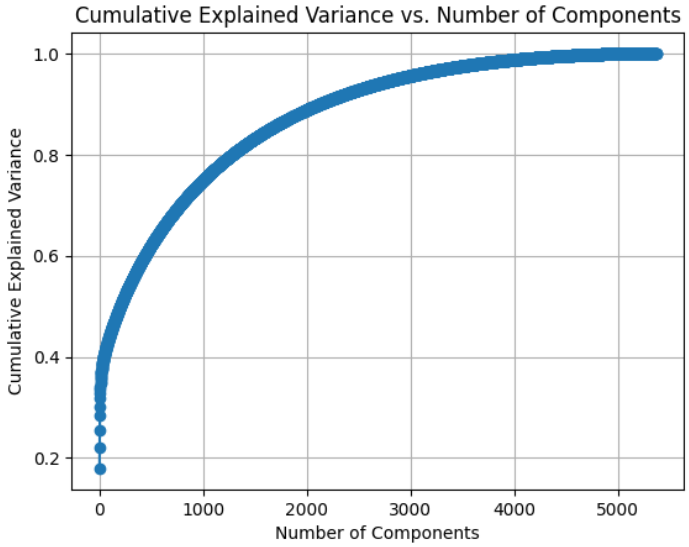


Figure 4-24 dataset components before PCA transformation

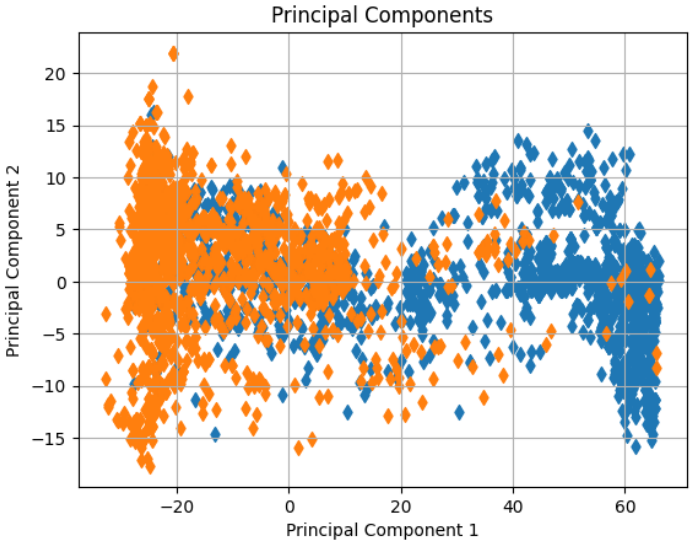


Figure 4-25 Shows the difference between 5 EGP and 10 EGP components value after PCA.

4.6.2.4 Fourth Stage of the: KNN Classification

1. Training the KNN Classifier: The KNN classifier "memorizes" the training data using extracted feature vectors, serving as a reference for predictions.

2. Choosing the Value of 'k': The 'k' value balances underfitting and overfitting, with cross-validation techniques helping to find the optimal value.

3. Distance Metric: KNN relies on distance metrics like Euclidean or Manhattan distance to compute similarity, impacting classification performance.

4. Classification: KNN calculates distances between feature vectors, identifies 'k' nearest neighbors, and assigns the input image to the majority class, recognizing currency denominations.

The model classifies currency images using KNN, accurately recognizing denominations and promoting financial independence for visually impaired users.

4.6.3 Design Constraints

1. Feature Integration: Combining ORB, histogram, and GLCM features requires careful integration, using concatenation or fusion techniques to preserve information content.

2. Feature Dimensionality: High-dimensional feature vectors from multiple feature types can increase computational complexity and memory requirements for KNN.

3. Feature Scaling: Properly scaling or normalizing features from different value ranges ensures equal contribution and prevents distance calculation biases.

4. Computational Complexity: KNN's complexity may be exacerbated when combining ORB, histogram, and GLCM features, demanding careful optimization.

5. Memory Requirements: Addressing increased memory usage with data compression, feature selection, or instance selection can accommodate devices with limited memory.

6. Robustness: Ensuring model robustness may involve fine-tuning feature extraction parameters, selecting an appropriate distance metric, and optimizing KNN's 'k' value.

4.7 Face Detection

The face detection algorithm used is based on the Viola-Jones algorithm. The reason this algorithm was used is that it is a very fast system, running at 14 frames per second, even though its accuracy is not the greatest.

4.7.1 Functional Description

The algorithm consists of mainly four steps:

1. Feature extraction: The algorithm uses Haar-like features, which are simple rectangular patterns that capture the contrast between adjacent regions in an image. These features are computed efficiently using an integral image representation.

2. Feature selection: The algorithm uses a machine learning technique called AdaBoost to select a small subset of features that are most relevant for face detection. This reduces the computational cost and improves the accuracy of the algorithm.

3. Classifier Construction: The algorithm builds a cascade of classifiers, each of which is composed of a linear combination of features selected by AdaBoost. The cascade is designed to reject non-face regions quickly, while passing face regions to the next stage.

4. Detection: The algorithm scans the input image at multiple scales and locations, applying the cascade of classifiers to each sub-window. If a sub-window passes all stages of the cascade, it is marked as a face region.

Each step will be discussed in detail in the next part.

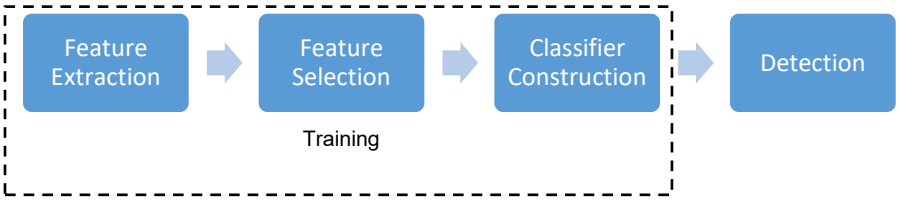


Figure 4-26 Face Detection Flow Chart

4.7.2 Modular Decomposition

As mentioned in the above part, the algorithm is based on four main steps. In each of the following paragraphs, we will discuss each step in detail.

4.7.2.1 Feature Extraction

The first step that needs to be done is creating the Haar-like feature that the algorithm will select from for face detection. Haar-like features are rectangular patterns that capture the contrast between adjacent regions in an image. The Haar-like features used are: two-

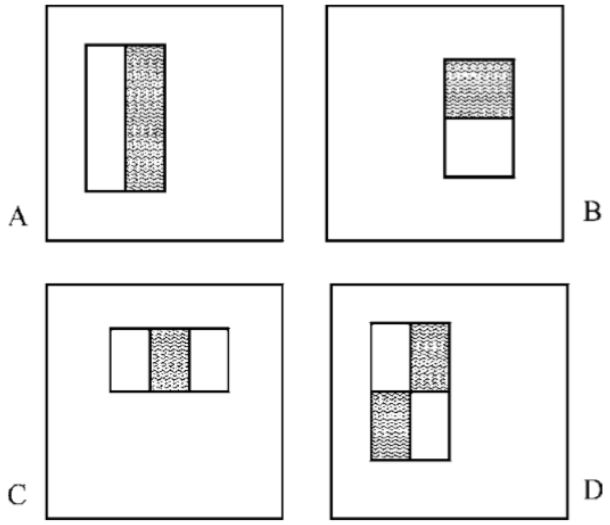


Figure 4-27 Examples of Harr-like features

rectangle, three-rectangle, and four-rectangle features. The two-rectangle features are two adjacent rectangles that either have the same height and different widths or have the same width but different heights. The three-rectangle features are similar to the two-rectangle features, but the difference is that there are three adjacent rectangles instead of two. The four-rectangle feature consists of four rectangles that are diagonal to each other. The features are created on a 24x24 image. This will create 162,336 features. Those features are saved and used for training.

4.7.2.2 Feature Selection and Classifier Construction

Since each one of those features is a weak classifier, Adaboost is used to select the most relevant Haar-like features and combine them into a strong classifier that can accurately detect faces in the image. The final output of the Adaboost is a list of weighted weak classifiers that can act as one strong classifier. This classifier is composed of twenty-five stages. A region in the image is considered a face if it passes all twenty-five stages. If one of the regions does not pass any of the stages, the region is immediately discarded and another region is considered. The reason for using stages is to reject non-face regions rapidly, thus reducing computation.

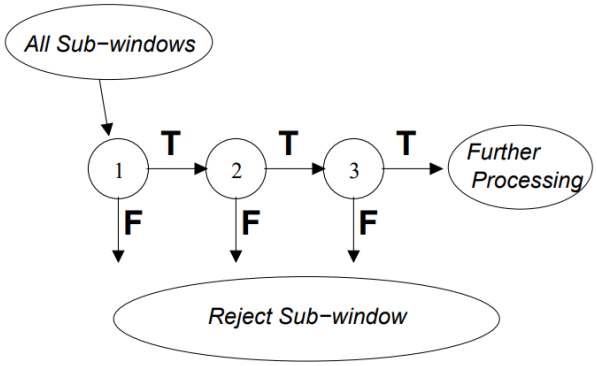


Figure 4-28 Example of cascaded classifier

4.7.2.3 Detection

After creating the cascaded classifier, the module is ready for face detection. The input image is first preprocessed. The preprocessing includes normalization and calculating the

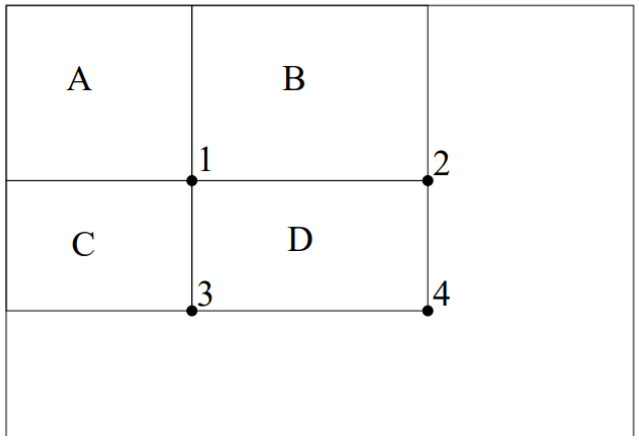


Figure 4-29 Example of an integral image. The sum of pixels in rectangle D equals to 4 - 3 - 2 + 1

integral image. An integral image is a data structure that allows for fast and efficient computation of sum of pixel values in a rectangular region of an image. It is also known as a summed-area table or a cumulative distribution table. The integral image is computed by adding up the pixel values along the rows and columns of the original image, such that each element of the integral image is equal to the sum of all the pixels above and to the left of it in the original image.

A sliding window is used to traverse the input image. Each sub-window is passed to the cascaded classifier, and if it passes all the stages, the region is considered a face. An image pyramid is formed by scaling down the input image by a scale factor. This is done to detect faces with different sizes.

4.7.3 Design Constraints

Even though the algorithm is a very rapid face detection algorithm, it has its drawbacks. The main drawback of the algorithm is that faces need to be upright and well illuminated in order to be detected.

Another constraint of the algorithm is that it needs a very large and labelled dataset of face and non-face images. This can be difficult because the non-face images should not contain any faces, so they need to be checked manually.

Finally, the training phase of the algorithm takes a considerable amount of time. This is caused by two things: the very large dataset and the fact that Adaboost cannot be run in parallel. One could argue that this is not a design constraint. However, if a modification is needed and retraining is required, this will be a very time-consuming process.

4.8 Emotion Detection

For individuals who are VIB, nonverbal cues such as facial expressions can be challenging to interpret, making it difficult to understand the emotions of others. Emotion detection technology has the potential to enhance the communication and social interactions of VIB individuals by providing them with a tool to better recognize and respond to the emotional states of others. By using advanced algorithms and machine learning techniques, emotion detection technology can analyze facial expressions to identify emotional states. This can help VIB individuals to better understand and respond to the emotions of others, leading to improved communication and stronger social connections.

4.8.1 Functional Description

The emotion detection module first starts by detecting any faces in the input image. If no faces are detected, the module is immediately terminated. In the event that a face is found, the module continues.

Once a face is detected, the module starts extracting facial landmarks from the face. Facial landmarks are specific points on a face that are used to identify and track various features and expressions. These landmarks are the (x, y) coordinates of key points on the face, such as the corners of the eyes, the tip of the nose, and the corners of the mouth. Here, the facial landmarks are used to analyze facial expressions.

After the extraction of facial landmarks from the face in the image, those landmarks are used to predict the emotion of the face. The prediction is done using a random forest classifier. The random forest classifier is trained on a labeled dataset of human faces expressing various emotions.

Figure 4-30 Emotion Detection Flow Chart

4.8.2 Modular Decomposition

The face detection part of the module uses the HOG feature descriptor. The HOG algorithm works by dividing an image into small regions called cells and computing a histogram of gradient directions for each cell. The histograms are then normalized and concatenated to form a feature vector that represents the shape and appearance of the image. Then, these descriptors are used with a linear classifier to detect faces.

The classifier is trained on a large dataset of face images and can detect human faces in a variety of lighting conditions and orientations. The model uses a sliding window approach to scan an input image at multiple scales and locations, looking for regions that contain facial features.

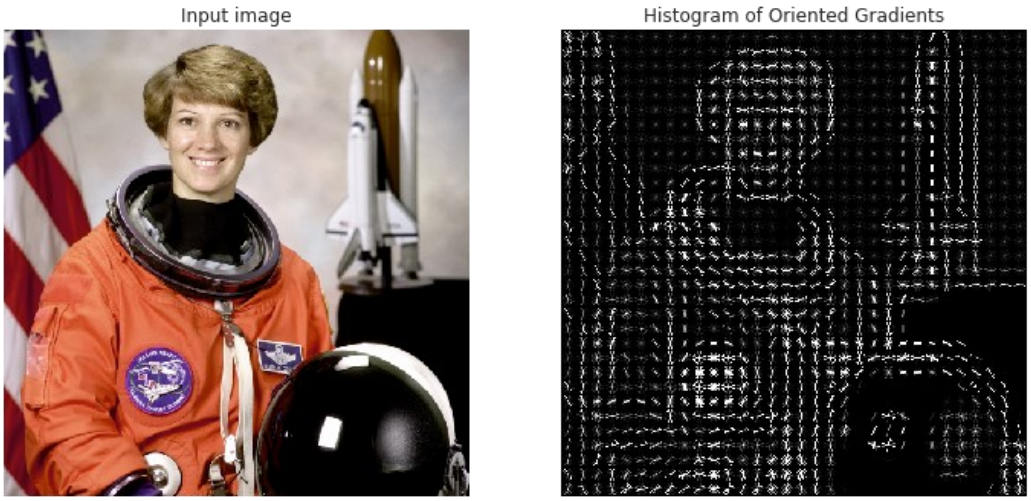


Figure 4-31 Example of HOG feature

As for the facial landmark extractor, the model uses a machine learning model that has been trained on a large dataset of annotated images to learn how to locate the landmarks. The model consists of an ensemble of regression trees that split the image into smaller regions and predict the offset of each landmark from the center of the region. The function combines the predictions of all the trees to obtain the final landmark coordinates.

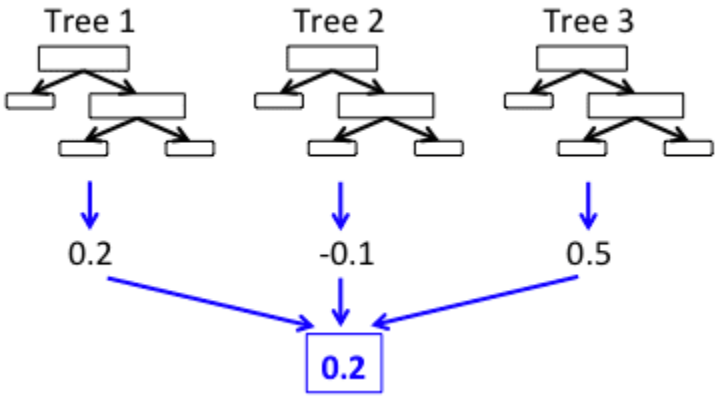


Figure 4-32 Example of ensemble of regression trees

Finally, emotion detection uses a random forest classifier to classify the emotion of the face based on the facial landmarks passed on from the previous stage.

4.8.3 Design Constraints

One of the main drawbacks of this design is that it cannot detect whether someone is suppressing their emotions. Moreover, detection of fake emotions is not possible.

4.9 Retail Product Identifier

Retail Product Identifier (RPI) is designed to aid VIB people in shopping. RPI uses the camera of a smartphone to scan the product and provide audio feedback about its name and brand. RPI aims to empower VIB people with more information and convenience when shopping for their needs and preferences.

4.9.1 Functional Description

The module consists of two main parts:

1. The first part is product detection. It detects the type of retail product. And returns a string that contains the type of products found. Then, it passes those product images to the next stage.

2. The second part is logo detection. Just like humans, it knows the brand of the retail product from the logo. It also returns a string that contains the logo detected for each item.

4.9.2 Modular Decomposition

The first part of the module that detects the product type is based on YOLO algorithm. YOLO is a state-of-the-art object detection algorithm that can detect and classify multiple objects in an image with high speed and accuracy. Unlike traditional object detection methods that use a sliding window approach or region proposal networks, YOLO divides

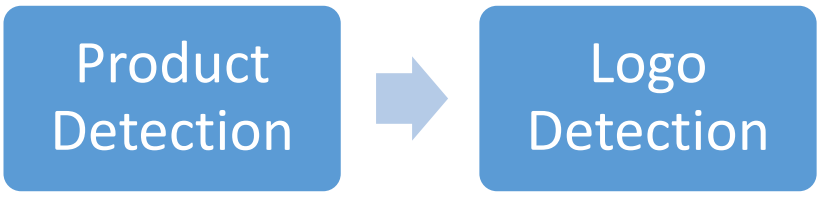


Figure 4-33 Flow Chart of Retail Product Identifier

the input image into a grid of cells and predicts bounding boxes and class probabilities for each cell. YOLO also uses a single neural network to perform the detection task, which makes it faster and more efficient than other methods that use multiple networks or stages.

The second stage is also based on YOLO architecture. As it proved to be superior in terms of speed and accuracy. Due to the lack of a sufficient dataset, data augmentation has been used on the dataset to compensate for that.

4.9.3 Design Constraints

Since YOLO is based on a neural network, it needs a very large dataset. This presented difficulties in the training process.

4.10 Apparel Recommender

The apparel recommender module for visually impaired or blind people is a technology that aims to improve independent living for visually impaired or blind individuals. This module uses computer vision and machine learning techniques to provide recommendations for clothing items that match the user's preferences or that match specific fashion rules.

4.10.1 Functional Description

An apparel recommender system is a type of recommender system that operates in the domain of garments and fashion products. It aims to provide personalized suggestions of apparel items or outfits to users based on their preferences. An apparel recommender system can use various types of information, such as texture, color, and clothing types. This information is gathered from the clothing descriptor module. These features are transformed into vectors, and a similarity matrix is formed, calculating the angle between each entry and the user's preference. Based on that angle, a recommendation is suggested.

Three actions can be taken in this module. The first is that the user can add some clothes recognized in the clothes descriptor module to their own wardrobe. The user can add the clothes to their preferences as well. There are also rules that define which clothes go best with which. Finally, the user is able to ask the module for apparel recommendations.

The module is able to save each user's wardrobe and preferences. This is done using the user's MAC address. This approach was used to save the user the hardship of signing up and logging in.

4.10.2 Modular Decomposition

The first thing the module does is vectorize the clothing features. This is done using TF-IDF. It is a statistical technique used in information gathering. TF-IDF is used to calculate the relevance of clothing items to a user's preferences based on the clothing features, which are texture, color and type.

After that, a cosine similarity matrix is formed. It measures the angle between the vectorized features and user preference.

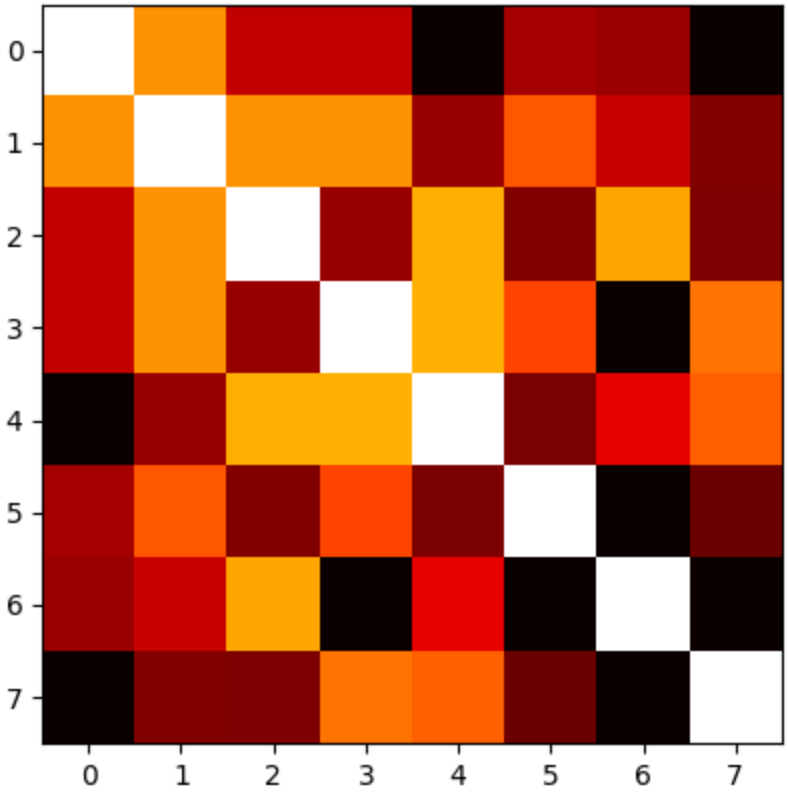


Figure 4-34 Example of the Cosine Similarity Matrix

Since the user is VIB, their preference may not be the best option in terms of fashion. That's why the system provides for some rules of fashion to be followed. For instance, it is not considered a good idea to wear shorts with a plaza on top; for that reason, no rule

defines it. On the other hand, wearing a blue T-shirt with black trousers is perfectly fine. So, a rule is provided stating that.

4.10.3 Design Constraints

Fashion recommendation is a huge system, and providing such rules to define it simplifies its complexity. Nonetheless, this means that the system becomes more constrained in terms of recommendations, as such rules are usually not enough.

4.11 Face Recognition using Eigenfaces.

The Eigenfaces algorithm is a technique for facial recognition that is based on the machine learning concept of Principal Component Analysis (PCA). It is one of the earliest techniques that have been developed for facial recognition, having been developed in the 1980s by Sirovich and Kirby. Since then, it has become one of the most popular techniques for facial recognition.

4.11.1 Functional Description

This module is designed to enable users to recognize the faces of their friends. It accepts an image of a person's face as input and provides the user with either the name of the person if it is recognized, or a message indicating that the person was not detected. Due to the way this algorithm works, which will be detailed in the following subsections, adding a new unknown face in the database for future recognitions would require training to be done on the whole dataset again, which is impractical. This is why we have taken the freedom of defining the inputs and outputs of the module as described above.

4.11.2 Modular Description

The basic idea behind the algorithm is to represent facial images as linear combinations of a small number of characteristic feature vectors, called “Eigenfaces”. These Eigenfaces represent the principal components, or Eigenvectors, of the distribution of facial images in a training set. We calculate them by performing Principal Component Analysis (PCA).

PCA is a mathematical feature extraction and selection technique that is often used in machine learning algorithms. The technique analyzes a set of training data and identifies the most important features, or “principal components”. These principal components

describe the variability in the dataset. In our case, these principal components represent the features or pixels that help the most in distinguishing between faces.

Now we will explain the working of the algorithm. First, the facial images are normalized by subtracting them from the average or mean face, which is also calculated from the training set. The images are then preprocessed by using smoothing filters. This helps with noise reduction and in enhancing the most important features.

Next, the normalized images are used to create the covariance matrix: it represents the statistical relationship between the different pixel values or features in the image. The covariance matrix is a square matrix that summarizes important relationships in the data. It captures the important data variations and provides insight as to how different features change together.

After calculating the covariance matrix, its Eigenvectors and Eigenvalues are calculated. These are calculated using linear algebra. The eigenvectors represent the directions in which the data varies the most, while the eigenvalues indicate the amount of variance along each eigenvector. Each Eigenvector corresponds to one Eigenvalue. Therefore, the Eigenvectors that have the highest Eigenvalues represent the directions in which data varies the most, which corresponds to the features that are best for differentiating between the different classes, or in our case faces. Based on some threshold, whether it be on the number of Eigenvectors or the minimum value of the variance, some Eigenvectors are selected while others are discarded.

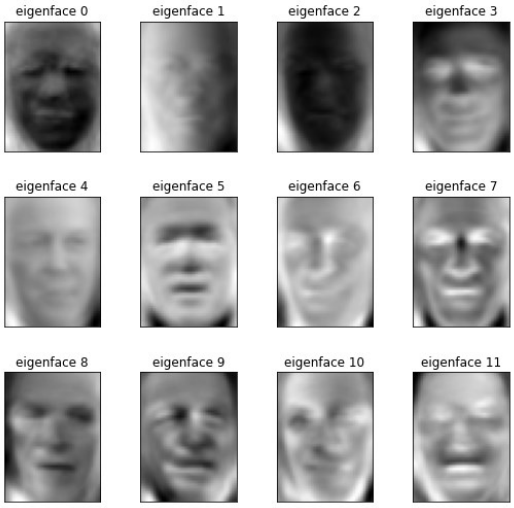


Figure 4-35 - Eigenfaces

All of what has been described is the steps taken in the training phase. To recognize a new face, the algorithm projects the face onto the Eigenfaces and calculates the distance between the projected face and the Eigenfaces in the dataset. The face is then

classified with the closest match. Any classifier at this point can be used. In our case, we used K-Nearest Neighbors (KNN) with K = 3.

As for the dataset, we used our own faces. Each of us captured some photos with different angles, expressions and different lighting conditions. We used these photos to train the algorithm and got impressive results: the algorithm recognizes us in most of our other testing photos. Measuring accuracies here would not be exactly meaningful because of the small size of the dataset. However, we have tried with another dataset, the Olivetti dataset, and got a 97% accuracy. We used 400 images of 40 people where each image is 64 x 64 pixels. We used 320 images for training and 80 images for testing.

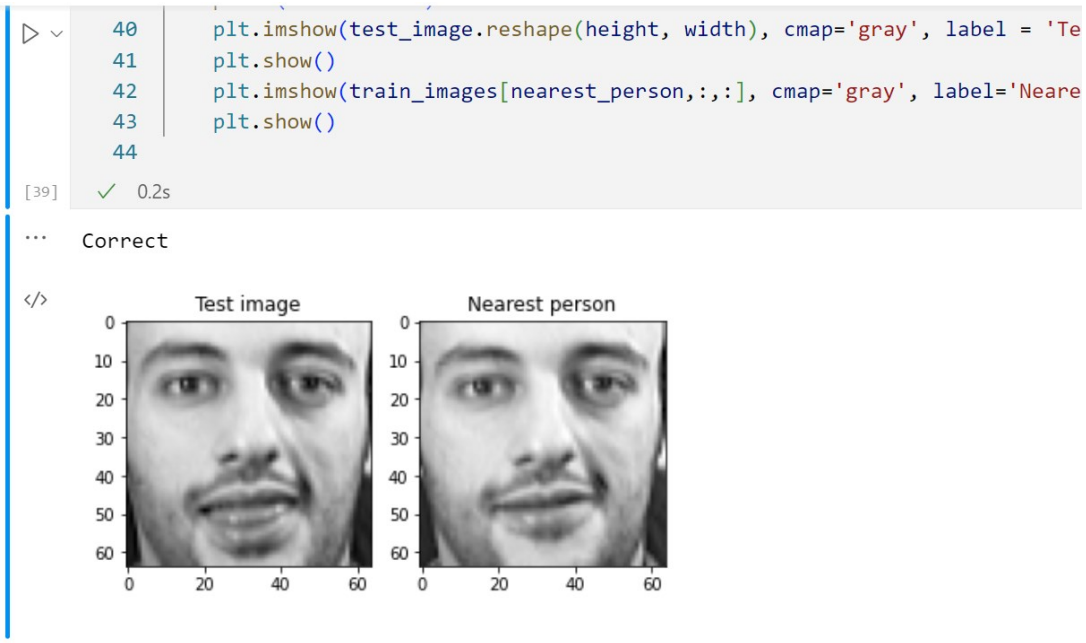


Figure 4-36 - Correct classification

One point of weakness of this algorithm is that the images must be close ups of faces. The algorithm therefore did not perform well for datasets where faces were not the main element in the image (15% accuracy for the LFW dataset).

This is expected of the Eigenfaces algorithm because it does not try to extract certain features from the images that would make the difference between one face and the other. Rather, it assumes that the faces distribution over the whole image space (which is all possible combinations of pixels ≡ all possible width x height images) is not random. Based on this assumption, the algorithm aims to calculate the Eigenvectors (called Eigenfaces) that best describe the distribution of face images over the images space.

4.12 Frontend

For the front-end part of our application, we used the Android development framework Flutter, developed by Google. This tool allows the building of high-performance, cross-platform mobile apps for Android and iOS. It uses the Dart programming language, which was also developed by Google, to build mobile apps with reactive styles. Our choice of the framework was determined by many factors including the ease of use of Flutter. One of its main features is hot reload, which allows developers to see changes in their code almost instantly without having to build the whole app again. This greatly speeds up the development process and makes it easier to iterate on designs and features.

4.12.1 Functional Description

This module aims to be the interface with the user, and so it should be easy to use and intuitive. The navigation inside the application should be entirely voice-based so that the application is easy to use for VIB individuals. It should also be fast and provide for real-time change between the different application modules.

A voice-based navigation means that the user can go from one module to the other by nothing other than voice commands. On the other hand, the application also needs to communicate information to the user, such as the output of our algorithms like the description of scenes or the classification of clothes and money denominations. This is why an equally important aspect must be present in the application: audio feedback.

Despite this, the UI is beautiful and allows for people whose visual impairment is not total blindness to enjoy a simple yet efficient experience.

Each module in the project corresponds to a Widget in the Flutter application, which maintains modularity, clarity, and ease of use. The users can customize the application's settings to adjust the rate of speech of the audio feedback.

The application also aims to be user-friendly even for users who have limited technology experience.

There are of course some limitations to the application. The application requires a stable and continuous access to internet. As unreliable as this may seem, a compromise must be made with the application's performance. Due to the very nature of AI and CV,

the algorithms must run on a highly efficient platform, such as a server with GPUs. To achieve this, constant connection to the internet must be made.

4.12.2 Modular Description

4.12.2.1 Startup Page

The Startup page is the first thing that a user may see when they open the application. It displays our logo and takes about a second before transitioning to the next page: the Homepage.

Figure 4-37 - Start Up Page

4.12.2.2 The Homepage

The Home Page is the central hub where users can navigate to all other modules. Upon loading, the home page will greet the user with a friendly message, providing a brief explanation of how to navigate the app. The greeting message is played as sound so that VIB individuals may not need assistance even in the first time they install the application. The greeting asks the user to speak any module name to navigate to it, only by voice command. On the background is displayed a semi-transparent image of our logo.

Users do not have to command the module name exactly to be directed to it. Instead, they can simply express their intent and the chatbot module Alan, which will be discussed further in this report, will understand what they want. For example, commands like ‘Scene Descriptor’, ‘Describe’, ‘Describe what’s in front of me’, ‘What is in front of me?’, ‘What can you see?’ and many more will direct the user to the Scene Descriptor module.

At this stage, the user can also directly speak with Alan. They can chat with Alan and ask for guidance about navigation or about other things. Examples to questions that users can ask Alan are: ‘What’s the weather like today?’, ‘Can you read this for me?’ and ‘How much is this?’.

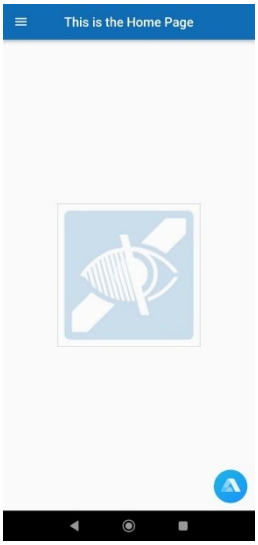


Figure 4-38 - Home Page

4.12.2.3 Side Bar Drawer

The user can open the menu drawer from the top left corner in order to navigate the application. However, as stated before, this is certainly not the only way to navigate the application. The main way is voice-based: the user directs navigation by voice commands only. However, for visually impaired individuals who are not completely blind and who may want to explore the different modules that we offer, or for VIB individuals who may have a sighted assistant, the menu drawer offers an extremely simple yet efficient overview of the different modules. It implements a beautiful and simple UI that lets any user get a coherent idea about what the application offers. Each module is represented by an expressive icon next to its name. For example, the scene descriptor has a camera, the face recognizes a face, the emotion recognizer a heart, the clothes descriptor a shirt and pants, and so on. Clicking on any one of them redirects the user to the appropriate module.

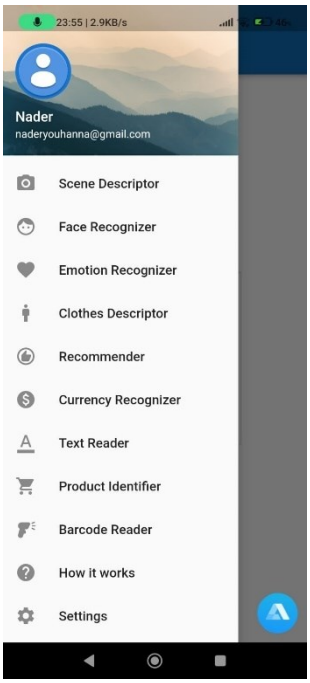


Figure 4-40 - Side Bar Drawer

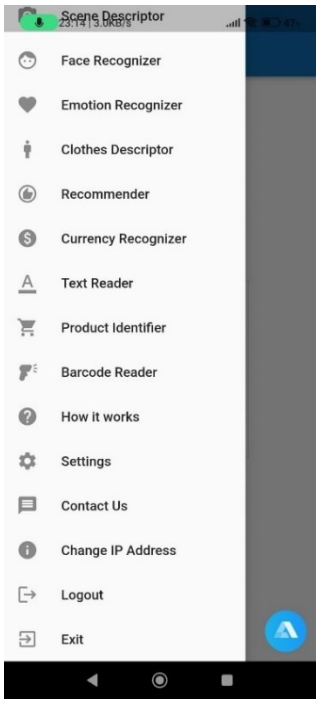


Figure 4-39 - Side Bar Drawer contd.

4.12.2.4 Scene Descriptor

When the user accesses the scene descriptor module, either through voice command or by clicking on the menu icon, a camera interface appears. Clicking on the camera button initiates a continuous loop that captures a photo, sends it to the server for analysis, and returns information about detected objects along with their distances from the device. This information is then spoken out loud to the user.

It is important to note that the loop will not start a new iteration until the previously spoken sentence is completed, regardless of its length. This process repeats indefinitely, allowing the user to continuously receive updated information about their surroundings.

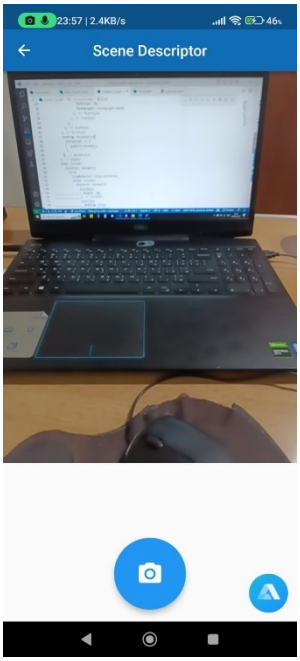


Figure 4-41 - Scene Descriptor Page

4.12.2.5 Other Modules

All the other modules in the application have the same form: a camera opens up and captures an image, which is sent to the server for analysis. Results are spoken out to the user. This includes Face Recognizer, Emotion Recognizer, Text Reader, Clothes Descriptor, and Currency Recognizer.

4.12.2.6 How It Works

The How It Works Module contains a brief paragraph that explains the basic functioning of the application, and which provides enough directions for any user to begin using the application. Since our target audience is VIB individuals, this message is not only displayed on the screen but also spoken out loud.

Chapter 5: System Testing and Verification

To test and verify the correct operation of each module, we have used various methods and tools, such as unit testing, integration testing and performance evaluation. In the following sections, we will describe each module in detail and explain how we have tested and verified its functionality and performance.

5.1 Testing Setup

Since the application is composed of different modules, the testing setup differs depending on the module. Each testing setup will be discussed in its own section.

5.2 Testing Plan and Strategy

Also, since the application is composed of different modules, the testing plan differs from one module to another. The testing plan and strategy will be discussed in its own section.

5.2.1 Module Testing

In each of the following sections, we will discuss how we tested each module as well as the result of the testing.

5.2.1.1 Face Detection

We first used the Olivetti dataset to test the viola jones algorithm. The dataset is composed of 400 images of faces. The accuracy was very high, 96%. When the module was tested on the LFW dataset, The accuracy was slightly lower at 93.2%.

5.2.1.2 Face Recognition

For our face recognition algorithm, we used our own faces as a dataset. We captured photos of ourselves with different angles, expressions, and lighting conditions, which we used to train the algorithm. The results were impressive, as the algorithm recognized us in most of our other testing photos. Since the dataset was small, we did not measure accuracies. However, we tested the algorithm with the Olivetti dataset, which comprises 400 images of 40 people, where each image is 64 x 64 pixels. We used 320 images for training and 80 images for testing and achieved 97% accuracy.

However, the algorithm has a weakness in that it requires close-ups of faces, and it did not perform well for datasets where faces were not the main element in the image. For example, the LFW dataset had only a 15% accuracy. This is expected of the Eigenfaces algorithm, as it does not extract specific features from the images that differentiate one face from another. Instead, it assumes that the distribution of face images over the image space is not random and calculates the Eigenvectors (Eigenfaces) that best describe this distribution.

5.2.1.3 Scene Descriptor

mAPval values are for single-model single-scale on COCO val2017 dataset. Speed averaged over COCO Val images using an Amazon EC2 P4d instance. We used YOLOv8m-seg for object detection in the scene description module. The mAP compares the ground-truth bounding box to the detected box and returns a score. The higher the score, the more accurate the model is in its detections.

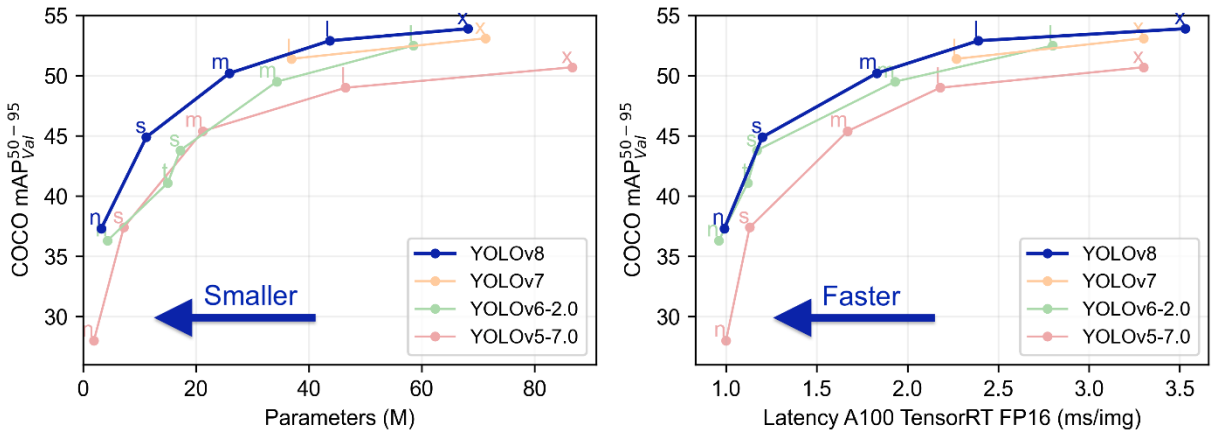


Figure 5-1 YOLO Versions Latency Comparison

Model	size (pixels)	mAP^{box} 50-95	mAP^{mask} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n-seg	640	36.7	30.5	96.1	1.21	3.4	12.6
YOLOv8s-seg	640	44.6	36.8	155.7	1.47	11.8	42.6
YOLOv8m-seg	640	49.9	40.8	317.0	2.18	27.3	110.2
YOLOv8l-seg	640	52.3	42.6	572.4	2.79	46.0	220.5
YOLOv8x-seg	640	53.4	43.4	712.1	4.02	71.8	344.1

Figure 5-2: YOLO segmentation Accuracy

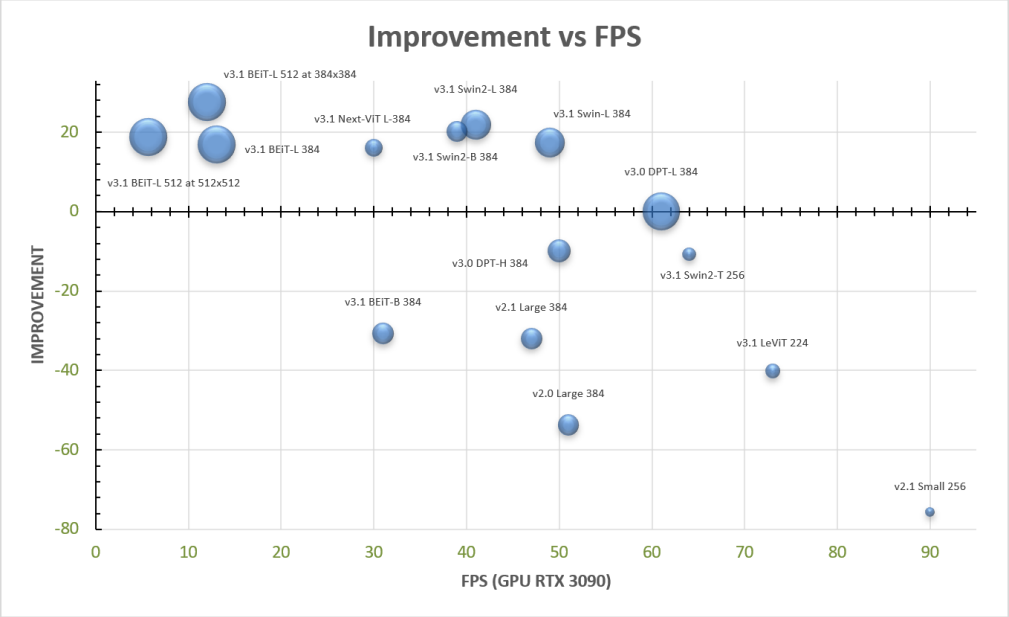


Figure 5-3 MIDAS Improvement vs FPS

MIDAS detects micro cluster anomalies up to 48% more accurately and as much as 644 times faster than current baseline approaches.

As Figure 4.8: Plotting Triangularization equation shows, the triangularization equation for distance estimation using depth values from heat map was not completely accurate, the additional error on the training dataset was the following:

$$R^2 = 0.6513$$

Where R is the sum squared regression

5.2.1.4 Clothes Descriptor

To validate the Mask R-CNN approach, we compared our computed bounding boxes to those provided by the COCO dataset. We found that ~2% of bounding boxes differed by 1px or more, ~0.05% differed by 5px or more, and only 0.01% differed by 10px or more.

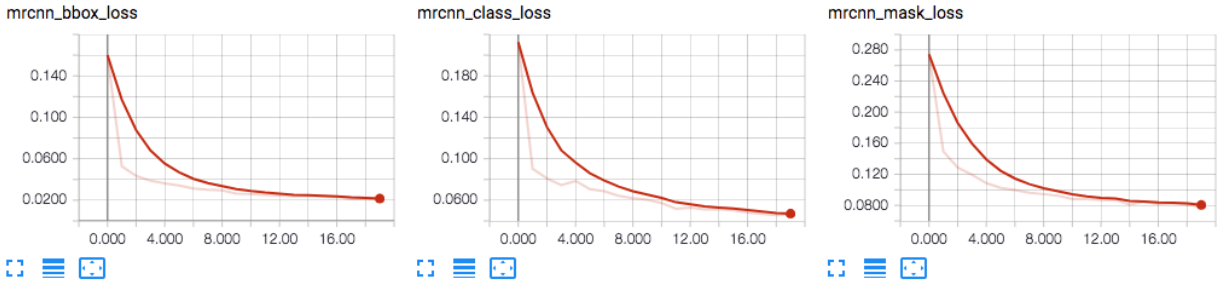


Figure 5-5 Training Losses Mask R-CNN

Model: SVM - Dataset: Train
Accuracy: 58.14%
Model: SVM - Dataset: Validation
Accuracy: 58.68%
==========
Model: KNN - Dataset: Train
Accuracy: 90.49%
Model: KNN - Dataset: Validation
Accuracy: 66.12%
==========
Model: Ensemble - Dataset: Train
Accuracy: 100.0%
Model: Ensemble - Dataset: Validation
Accuracy: 94.21%
==========
Model: AdaBoost - Dataset: Train
Accuracy: 59.05%
Model: AdaBoost - Dataset: Validation
Accuracy: 55.37%
==========
Model: ANN - Dataset: Train
Accuracy: 97.81%
Model: ANN - Dataset: Validation
Accuracy: 90.91%

.

These metrics allow us to assess the performance of each model and compare their effectiveness in accurately classifying textures. However, for our comparison, we mainly focused on the accuracy metric, as it provides a comprehensive measure of overall classification performance.

After training, the random forest model appeared to have the highest accuracy for texture detection, and it was saved as a .pkl file for future use.

5.2.1.5 Text Reader

After preprocessing the EMNIST dataset and training the model, the accuracy achieved after 25 iterations was 93.5%. This is because some letters have similar shapes, which can be confusing, such as "v" and "r" or "g" and "q". The figure below illustrates how the shapes of these letters can be misleading, causing the model to struggle in distinguishing them accurately. Nonetheless, an accuracy of 93.5% indicates that the model performs well overall in recognizing and classifying the majority of characters in the dataset.

Figure 5-6 Testing Text Reader Model

5.2.1.6 Currency Detector

The k-Nearest Neighbors (k-NN) model classifies data based on a mix of texture features and SIFT (Scale-Invariant Feature Transform) features. Before using the k-NN model, they reduced the data size with Principal Component Analysis (PCA) while keeping the important information. This combination of steps and the k-NN model led to a high accuracy of 94.3%. This shows that using both texture and SIFT features can help make classification tasks more accurate.

5.2.1.7 Product Identifier

During the training of the module, the accuracy was calculated using the testing set. The original dataset was divided into a training dataset comprising 85% of the data and a testing dataset comprising the remaining 15%. The accuracy of the modules was evaluated using the testing dataset, and the results showed an accuracy of approximately 70%. This approach allowed for the assessment of the modules' performance and ensured that the module was somewhat accurately identifying the products.

5.2.1.8 Apparel Recommender

Testing the apparel recommender system proved to be a challenging task. As a result, the system was tested using user feedback as the primary evaluation metric. Specifically, users were asked to provide feedback on whether they liked the recommended outfits or not. Overall, the user feedback was positive, indicating that the system provided satisfactory recommendations. Despite the difficulty in testing the system, the user feedback helped ensure that the apparel recommender system performed as intended and met the needs of its users.

5.2.1.9 Emotion Detection

The dataset that was used is Cohn-Kanade (CK and CK+) Dataset. These datasets were created by researchers at the University of Pittsburgh and contain a large number of images and corresponding labels of facial expressions. The CK dataset contains 486 image sequences of 97 subjects, with each sequence showing a series of facial expressions from neutral to full expression.

Around 20% of the dataset was used for testing. The detection rate for 8 emotions was approximately 77%.

5.2.2 Integration Testing

After testing each module separately, we performed manual testing to verify the integrated application's correct operation. Firstly, we tested the communication between the frontend and backend to ensure that images were sent successfully. Once we had verified successful communication, we proceeded to test the application as a whole. This involved running the application and ensuring that all modules worked together seamlessly to provide the intended functionality. Through this manual testing process, we were able to identify any defects or issues that may have arisen during the integration of the modules and ensure that the application was reliable and accurate.

5.3 Testing Schedule

In the testing schedule for this project, a continuous testing approach was adopted where every implemented module was immediately tested. This approach ensured that any defects or issues were identified and resolved quickly, reducing the risk of defects being carried forward to subsequent stages of development. Continuous testing also enabled the development team to receive timely feedback on the quality of their work, allowing them to make necessary adjustments and improvements as needed. By

implementing continuous testing, the project was able to undergo thorough and effective testing throughout the development lifecycle, resulting in a higher quality end-product that met the needs and our expectations.

5.4 Comparative Results to Previous Work

Table 5-1: Competitive Products

Features \ Apps	Lookout	Envision	Soundscape	Facing Emotions
Chatbot	X	X	X	X
Scene Descriptor	Done	X	Done	X
Text reader	Done	Done	X	X
Emotion detector	X	Done	X	Done
Currency detector	Done	X	X	X
Products detector	Done	X	X	X
Clothes Descriptor	X	X	X	X
Clothes Recommender	X	X	X	X

The table presented here displays the various modules incorporated in our application and indicates whether they are present in competitor products. Our application comprises eight main modules: Chatbot, Scene Descriptor, Text Reader, Emotion Detector, Currency Detector, Products Detector, Clothes Descriptor, and Clothes Recommender. While some of these modules can be found in competitor products, there is no all-encompassing, generic application in the market that covers all aspects of visually impaired individuals' daily lives.

Chapter 6: Conclusions and Future Work

The development of machine learning applications in a real-world setting can be a challenging task. From limited computational power to finding suitable datasets and obtaining feedback from users, there are various obstacles that developers must overcome to create reliable and effective systems. In this chapter, we discuss the challenges we encountered during the development of a system for aiding visually impaired and blind individuals. We highlight the main challenges we faced, including limited computational power and lack of diversity in datasets, and describe how we overcame these challenges to create a reliable and effective system. We also discussed our experience in using cloud-based machine learning frameworks and selecting appropriate datasets for the application. Through this chapter, we aim to provide valuable insights into the challenges and opportunities of developing machine learning applications in a real-world setting.

6.1 Faced Challenges

We encountered several challenges during the development of the system, with one of the main challenges being the limited computational power. The datasets we used were very large and required a significant amount of computation, which posed a challenge for the development team.

Another challenge we faced was the lack of diversity in the datasets we were able to obtain. This resulted in relatively poor performance when applying the module to real-world scenarios. Despite this challenge, the team worked to optimize the system to the best of their ability, and the results obtained with the available datasets provided valuable insights for further improvements.

Obtaining feedback from visually impaired and blind (VIB) individuals was a challenging task due to the limited means of reaching out to them. As a result, we faced scarcity in obtaining feedback from this group. Despite efforts to find efficient ways to reach out to VIB individuals, it remained a challenge throughout the development process.

6.2 Gained Experience

Our experience during the development of the system was mainly focused on using cloud-based machine learning frameworks, particularly Google Colab. This platform allowed us to leverage the power of cloud computing and access powerful machine learning tools to develop and test the system.

Another area where we gained valuable experience was in selecting and filtering datasets. We had to assess various datasets to find the most suitable ones that matched our requirements. This process involved careful consideration of the dataset's size, diversity, and relevance to our application. Through this experience, we gained a deeper understanding of the importance of selecting appropriate datasets for machine learning applications.

Overall, the development of the system provided us with valuable insights into the challenges and opportunities of developing machine learning applications in a real-world setting. We were able to leverage cloud computing tools, utilize suitable datasets, and incorporate feedback to create a reliable and effective system for aiding visually impaired and blind individuals.

6.3 Conclusions

Developing a machine learning system for visually impaired individuals was challenging but rewarding. Despite obstacles like limited computing power and dataset diversity, we created a reliable tool through optimization and user feedback. Our experience emphasized the importance of considering real-world challenges when developing machine learning applications.

6.4 Future Work

Going forward, we hope that our experience and insights can help guide future developments in this field and contribute to the creation of more effective and accessible tools for individuals with disabilities.

There are several areas for potential improvement in the developed application. One such area is the product identifier module, as accurately recognizing retail products is essential for visually impaired and blind (VIB) users. Improving the accuracy and reliability of this module would greatly enhance the application's usability and value for users.

Additionally, expanding the application's functionality to include a complete virtual assistant would be a significant improvement. This would allow users to access a broader range of features and services, such as voice commands, navigation, and other helpful tools. By incorporating these additional features, the application could become a more comprehensive and valuable tool for assisting VIB individuals in their daily lives.

References

[1] T. Diwan, G. Anirudh and J. V. Tembhurne, "Object detection using YOLO: challenges, architectural," *Multimedia Tools and Applications,* vol. 82:9243–9275, 8 August 2022.

[2] S. M. H. Miangoleh, S. Dille, L. Mai, S. Paris and Y. Aksoy, "Boosting Monocular Depth Estimation Models to High-Resolution via Content-Adaptive Multi-Resolution Merging," *CVPR,* Vols. 9685-9694, 2021.

[3] Y. Ge, R. Zhang, X. Wang, X. Tang and P. Luo, "DeepFashion2: A Versatile Benchmark for Detection, Pose Estimation,," *CVPR,* 2019.

[4] B. Xin, J. Zhang, R. Zhang and X. Wu, "Color texture classification of yarn-dyed," *Textile Research Journal,* vol. 87(15) 1883–1895, 2017.

[5] P. Viola and M. Jones, "Rapid Object Detection using a Boosted Cascade of Simple," *International Conference on Computer Vision,* 2001.

[6] M. Turk and A. Pentland, "Face Recognition Using Eigenfaces," *Journal of Cognitive Neuroscience,* 1991.

[7] M. H. Kim, Y. H. Joo and J. B. Park, "Emotion Detection Algorithm Using Frontal Face Image," *Institute of Control, Robotics and Systems,* 2005.

[8] Y. Wei, S. Tran, S. Xu, B. Kang and M. Springer, "Deep Learning for Retail Product Recognition: Challenges and Techniques," *Computational Intelligence and Neuroscience,* 2020.

[9] X. Yang, S. Yuan and Y. Tian, "Assistive Clothing Pattern Recognition for Visually Impaired People," *IEEE,* 2014.

[10] P. V. Jamkhandikar D, "Indian Currency Recognition System," pp. 2395-566, 2021 .

[11] Y. W. Zhang Q, "Currency Detection and Recognition Based on Deep Learning," *15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS),* pp. 1-6, 2018.

[12] T. M. Aqab S, "Handwriting Recognition using Artificial Intelligence Neural Network and Image Processing," 2020.

[13] R. A. Beohar D, "Handwritten Digit Recognition of MNIST dataset using Deep Learning state-of-the-art Artificial Neural Network (ANN) and Convolutional Neural Network (CNN)," *International Conference on Emerging Smart Computing and Informatics (ESCI) ,* pp. 542-548, 2021.

Appendix A: Development Platforms and Tools

This appendix provides an overview of the development platforms and tools used in this project, covering everything from integrated development environments (IDEs) to source control management systems, and more.

A.1. Hardware Platforms

For the server, we used an Intel i7 9th generation processor and Nvidia GTX 1660 Ti GPU with 16GB of Ram. As for the mobile device, we used an Android 13 phone with 6 GB of Ram.

A.2. Software Tools

The software that was used for the development process:

1. Flutter
   Flutter is a mobile app development framework that enables the creation of high-performance, cross-platform apps for both Android and iOS platforms. It leverages the Dart programming language and employs reactive styles to build mobile applications. Our decision to use Flutter as the framework for our project was influenced by various factors, including its user-friendly nature and ease-of-use. It was used to develop the mobile application that the user will use.
2. Python
   Python is a high-level, interpreted programming language that is widely used for general-purpose programming, web development, data analysis, artificial intelligence, and scientific computing. Python was used to build the backend that would run on the server. It was also used to build all the modules in the project.
3. VSCode
   Visual Studio Code is a free and open-source code editor developed by Microsoft. It is a popular choice among developers for its extensive range of features and support for a wide variety of programming languages. It was used as the Integrated development environment for building the frontend, backend and the AI modules.
4. git
   Git is a popular and widely used distributed version control system used for source code management. It was used as a control system and GitHub was used to host the repository to act as project control flow tool.

Appendix B: Use Cases

The following describes the use cases. In all use cases, the first thing a user will encounter after opening the application is the Home Page, from which they can speak any module name to navigate to it.

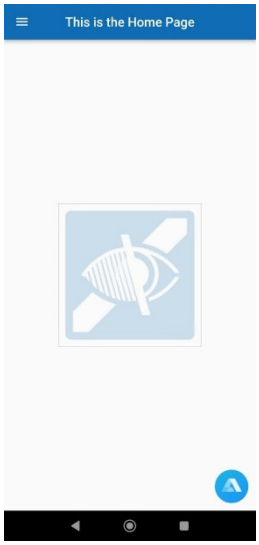


Figure 6-1 - The Home Page

Use case 1: Scene Descriptor

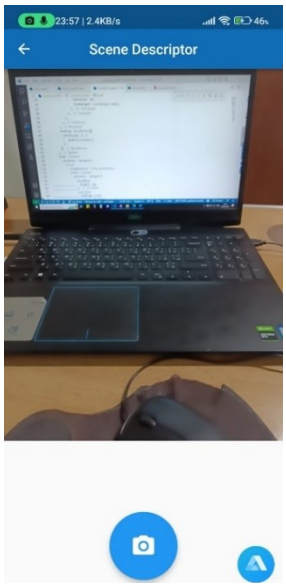


Figure 6-2 - Scene Descriptor

The application allows the user to speak a module name, such as "Describe", which redirects them to the Scene Descriptor module. The app captures photos and describes objects in the scene by voice, while estimating the distance from the user. This loop continues until the user says "Back" or "Stop", at which point the loop halts.

Use case 2: Clothes Descriptor
The user can choose to navigate to the Clothes Descriptor module by speaking a phrase such as "What is he wearing?" from anywhere in the application. The module would ideally output a description of the clothing in the photo, such as "This person is wearing a blue t-shirt, a watch, shoes and light grey cotton trousers."

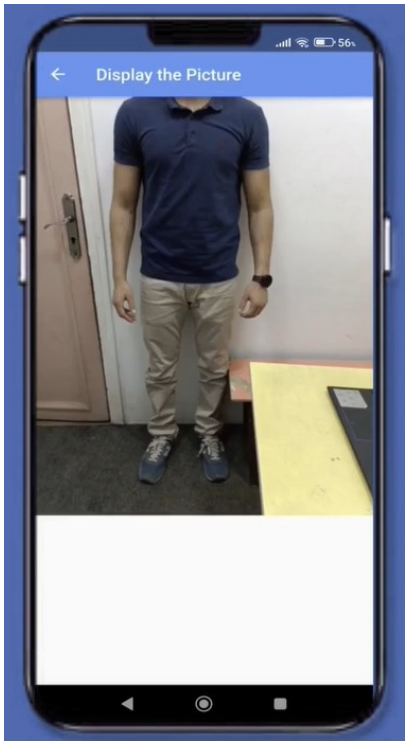


Figure 6-3 - Clothes Descriptor

Use Case 3: Emotion Recognizer

The user could also navigate to, for example, the emotion detection module. As the case with all other modules, this is easily done by just speaking ‘What are they feeling ?’.

Note that the user can speak this sentence from anywhere in the application and they will be succesfully redirected to the desired module.

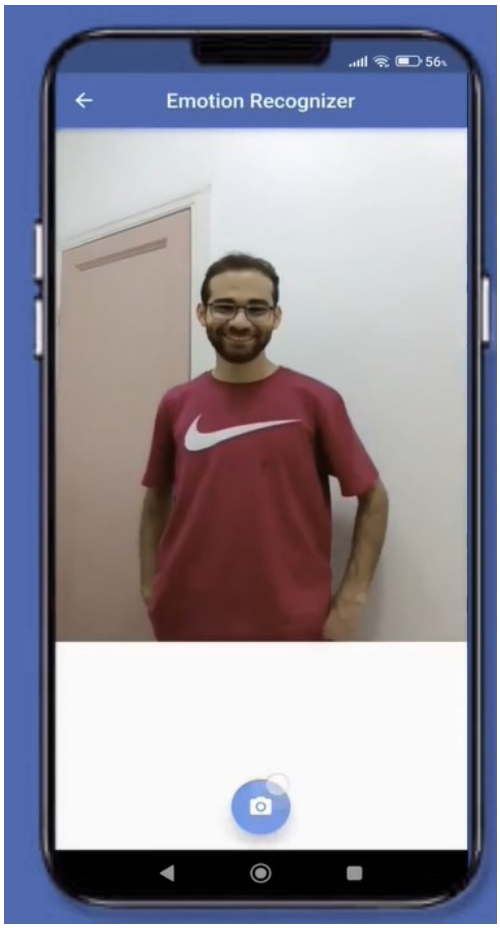


Figure 6-4 - Emotion Recognizer

Taking such a photo, for example, would result in the output ‘This person is feeling happy’.

Use Case 4: Currency Recognizer

By saying ‘How much is this ?’ or any variant of this sentence with the same meaning, the user will be redirected to the currency recognizer module, again, from anywhere in the application.

Figure 0-5 - Currency Recognizer

The application will also, as in the case with all other modules, begin to capture images and describe accordingly.

Use Case 5: Recommender

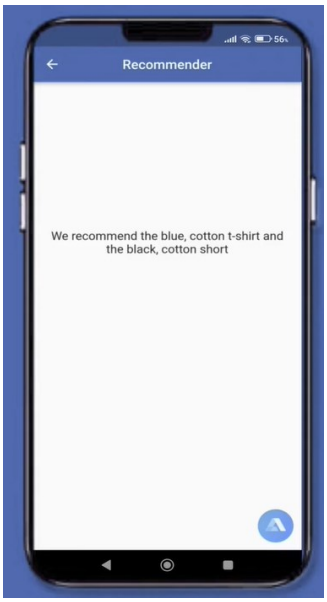


Figure 6-6 - Recommender

Use Case 6: Face Recognizer

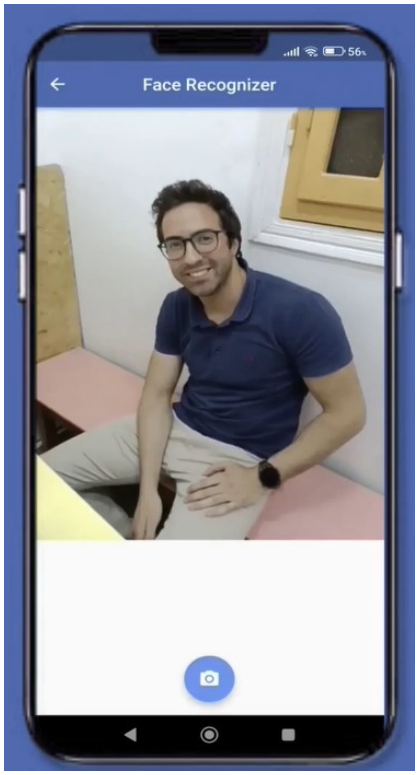


Figure 06-7 - Face Recognizer

Use Case 7: Text Reader

Figure 06-8 - Text Reader

Use Case 8: Product Identifier

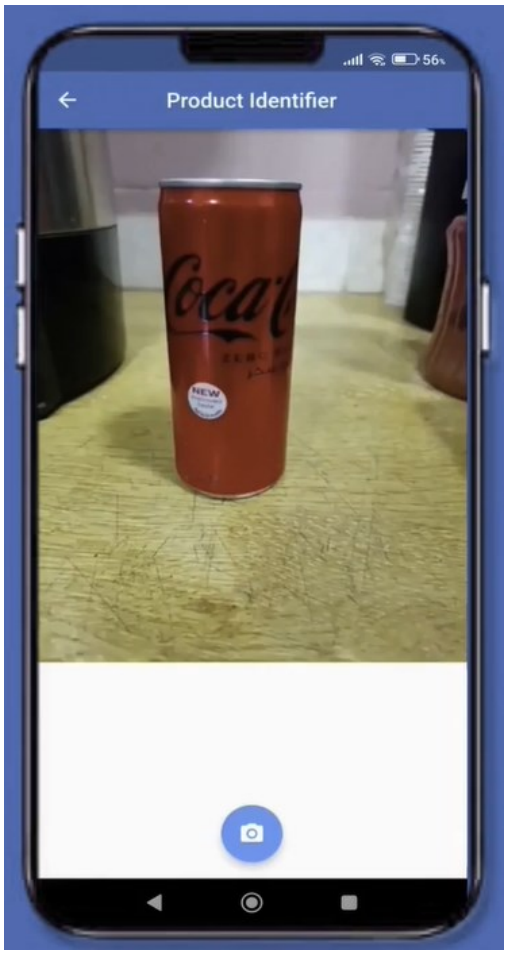


Figure 6-9 - Product Identifier

Appendix C: User Guide

To use our product, the user should install the application and open it. Upon opening it, they will be encountered with the Home Page. Then, they will be able to navigate by speaking the module name or any related sentence. For example, saying ‘Describe’ or ‘What is in front of me?’ or ‘What do you see?’ or any variant of this sentence with a similar meaning will redirect the user to the Scene Descriptor.

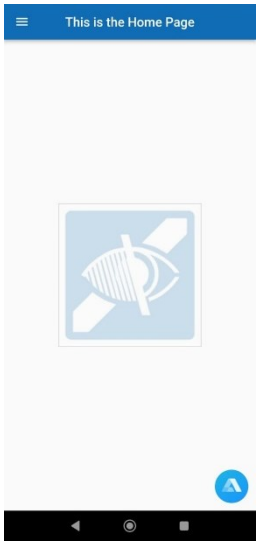


Figure 6--10 - Home Page

This way, the user can easily navigate the application.

Appendix D: Feasibility Study

Technical Feasibility: The project demonstrates strong technical feasibility. The structured approach based on software engineering principles ensures a reliable and effective solution. The selection of modules such as Scene Descriptor, Clothes Descriptor, Face Detector, Face Recognizer, Emotion Detector, Currency Recognizer, and Text Reader aligns with the identified needs of VIB individuals. The use of reliable coding practices and project flow control tools like git minimizes errors and ensures high-quality code.

Operational Feasibility: The mobile application holds promising operational feasibility. By addressing the challenges faced by VIB individuals through the selected modules, the application has the potential to significantly enhance their understanding of the environment and daily activities. A thorough understanding of user requirements and specifications ensures the application is designed with their needs in mind, making it user-friendly and intuitive.

Economic Feasibility: The economic feasibility of the project is favorable. The demand for mobile applications catering to the needs of visually impaired individuals is growing. This indicates a potential market for the application. Additionally, the project's reliance on efficient coding practices and available software tools helps control development costs. A cost-benefit analysis should be conducted to evaluate the project's financial viability further.

Scheduling Feasibility: The project demonstrates reasonable scheduling feasibility. The structured approach based on software engineering principles provides a clear roadmap for development. The sequential implementation of modules allows for focused development and testing, ensuring each component is thoroughly evaluated. However, it is crucial to establish realistic timelines and allocate sufficient resources to meet the project's objectives within the desired timeframe.

Quality Assurance: Rigorous quality assurance processes have been incorporated into the project. Comprehensive testing, including functionality, performance, and security testing, will be conducted to ensure the application functions are as intended and are robust and secure. Regular reviews and feedback from visually impaired users will also be valuable in identifying areas for improvement and enhancing the overall quality of the application.