

**Csci490 – Information System Development (Report)**

BE MY EYES

An Object Recognition and Spelling Application

**Prepared by: Instructor:**

Ahmad Jaber Dr. Ali Chouman

Jaber Moubarak

Spring 23-24

Chapter 1 System Requirements and Technical Specifications

1. Introduction to BeMyEyes……………………….…………………….…………………….…………………………………………3

2. Detailed description and Features…………………………………………………….………………………………….…………4

3. System architecture…………….…………………….…………………….…………………….…………………………………….…5

4. Flowchart diagram ….…………………….…………………….…………………….…………………….………………………..…..6

5. software and hardware requirements ……………….………………………….…………………….…………………..……7

6.Benefits ………….…………………….………………………….…………………….…………………………………………………..…..9

Chapter 2 Object Recognition Overview

1.Introduction to object recognition……………………….…………………….…………………….…………………………....11

2. Object localization……………..…………………….…………………….…………………………………………………………...…11

3. Depp Learning……………………………………………………..…………………………………………………………………………12

4. R-CNN………….…………………….…………………….…………………….…………………….…………………….……………….…13

4.1 Fast R-CNN…………….…………………….…………………….…………………….…………………….………………………..13

4.2 Faster R-CNN…………………….…….…………………….…………………….…………………….……………………………..13

5. YOLO Algorithm ….…………………….…………………….…………………….…………………….…………………………………14

5.1 The YOLO Architecture …………………….…………………….…………………….…………………….……………………14

5.2 ANCHOR BOXES…………………….…………………….…………………….…………………….………………………………..…15

6. MobileNet ………….…………………….………………………….…………………….……………………………………………….15

6.1 MobileNet Depth wise Separable Convolution ………….…………………….……………………………………..16

6.2 Advantages of MobileNet………….…………………….………………………….…………………….…………………….17

7. Conclusion ………….…………………….………………………….…………………….………………………………………………17

Chapter 3 Implementation and Functionality Overview

1. Introduction…………………………….……………………………………...…………………………………………...…17

1.1 implementation process.…………………………………………………………………….……………………...17

1.2 Technologies and tools used ………………………………………………………………………………………17

2. User Experience Design……………………….…………………………………………………………………...…….18

3. Enhancing Flutter App Quality with Widgets ……………………………………………………………………19

3.1 Optimizing Code Quality……………………………………………………………………………………………19

3.2 Harnessing Widgets……………………………………………………………………………………………………19

Chapter 1 System Requirements and Technical Specifications

1.Introduction to “Be my eyes”



Be My Eyes is an innovative application designed to empower individuals with visual impairments by providing real-time assistance through their smartphones.

Our application designed to enhance accessibility for visually impaired individuals, leverages cutting-edge technologies to provide real-time object detection and recognition. Through the integration of advanced machine learning models, such as YOLOv5 and MobileNet, our app offers users the ability to interpret their surroundings with unprecedented accuracy. By harnessing the power of computer vision and speech synthesis, our solution empowers users to navigate their environment independently, facilitating greater autonomy and inclusivity in their daily lives.

2.Detailed description and features



**To get started, simply tap on "Get Started" signaling your agreement with our privacy policy and unlocking a host of benefits. By agreeing, you ensure the security of your data while enjoying seamless access to our app's features.**

# When the button is turned off:

**Upper Section - Live View:**

**Experience real-time visuals of your environment with the live feed displayed at the top of the screen. Stay informed about your surroundings effortlessly.**

**Lower Section - Camera Control Button:**

**Located at the screen's bottom, this button offers control over the camera. By toggling it, users can start or stop the live broadcast. Feel the tactile response upon activation for added assurance.**

# When the button is turned on:

**Upper Section - Live View:**

This section showcases the live feed captured by the camera, occupying the top portion of the screen. Users gain real-time insight into their surroundings through this live broadcast.

**Lower Section - Camera Control Button:**

Strategically positioned at the screen's bottom, this button empowers users to control the camera. Activate it to initiate or cease the live broadcast, accompanied by tactile feedback for a seamless experience.

The lower part also contains a section for the text description of objects

**Features:**

Object Recognition:

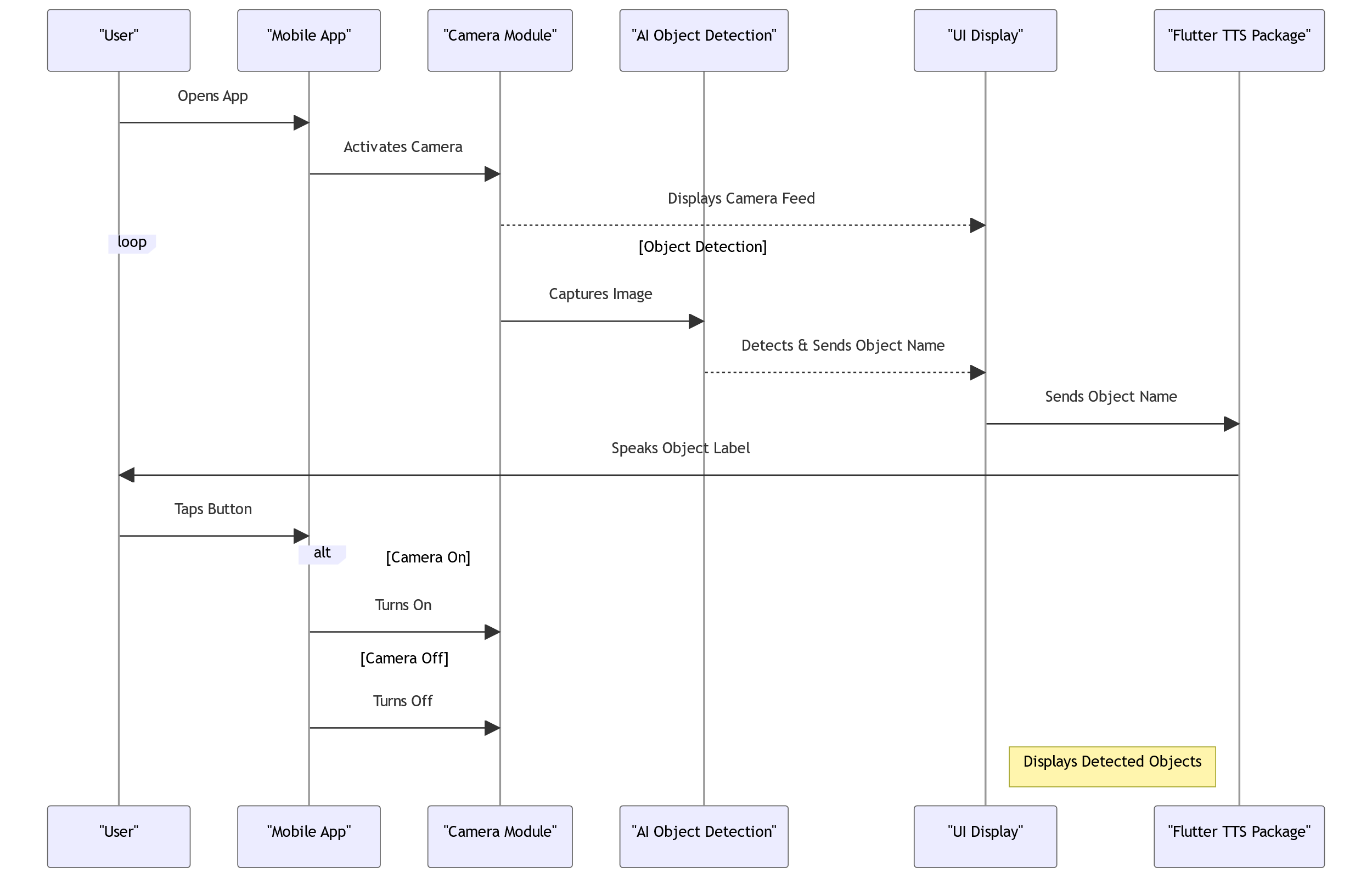
Central to the app's functionality is real-time object recognition. By leveraging the camera feed, the app swiftly identifies objects within its field of view. From commonplace items like chairs and tables to human figures and doors, detected objects are promptly announced to the user.

Audio Feedback:

The app offers comprehensive audio feedback to accompany various user actions, including camera activation/deactivation and object recognition. Through spoken cues, users receive clear indications of the app's status and ongoing operations.

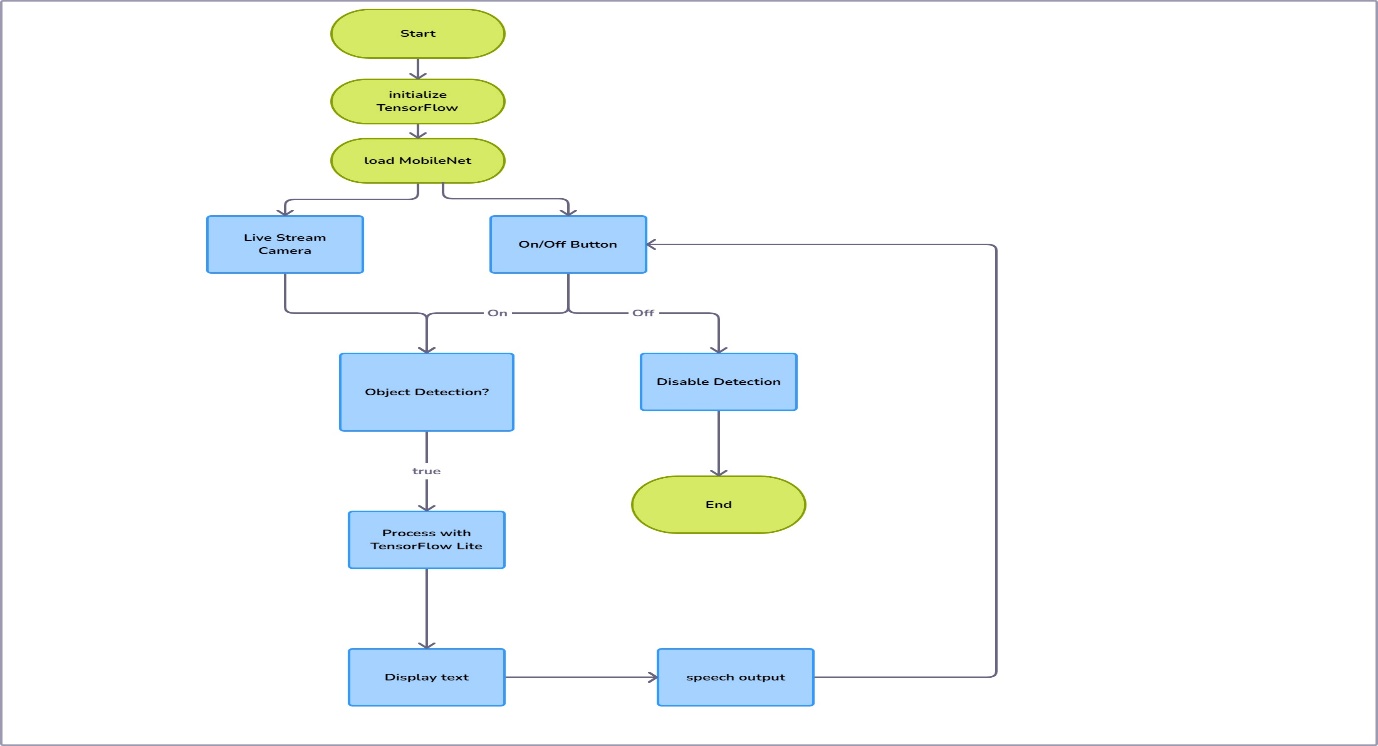
3.System architecture

The system architecture diagram provides a visual representation of the project's structural framework, aiding in understanding the flow of information and processes.



The app utilizes the camera of a mobile device to capture live video streams of the user's surroundings. These video feeds are processed in real-time using state-of-the-art object detection and recognition models, such as YOLOv5 and MobileNet, implemented through TensorFlow. As the app analyzes the video frames, it identifies and labels various objects present in the environment. These labels are then converted into speech feedback using text-to-speech synthesis technology, providing auditory cues to the user about their surroundings. Additionally, the app offers features such as toggling the camera, accessing live streams, and customizing speech feedback settings, all through an intuitive user interface. This seamless integration of computer vision and speech synthesis technologies allows users to navigate and interact with their surroundings effectively, enhancing their independence and accessibility.

4.flow chart diagram



Upon launching the application, TensorFlow is initialized to enable object detection. Subsequently, the MobileNet model is loaded to support real-time object recognition. If the detection button is toggled off, object detection ceases, and the process concludes.

However, if the button remains switched on, the application retrieves frames from the camera and processes them using TensorFlow Lite. When an object is detected, its label is displayed in the lower section of the interface. Additionally, a voice speech is generated to audibly announce the identified object, ensuring accessibility for visually impaired users.

5.software and hardware requirement

Software Requirements:

* Flutter SDK: Essential for developing cross-platform mobile applications using the Dart programming language.
* Android Studio: Integrated development environments (IDEs) for Flutter app development.
* Dart SDK: Required for compiling Dart code to native machine code.
* Mobile Device Emulators: Tools for testing apps on virtual devices during development.
* Text Editor or IDE Plugins: Additional plugins for enhancing the development experience in the chosen text editor or IDE.
* Flutter Dependencies: External packages and libraries used to extend the functionality of Flutter apps.
* TensorFlow Lite: Machine learning framework for running lightweight models on mobile and IoT devices.
* TFLite Plugin: Integration plugin for TensorFlow Lite models in Flutter applications.
* Permission Handler Plugin: Plugin for managing runtime permissions in Flutter apps.
* Audio Player Plugin: Plugin for playing audio files in Flutter apps.
* GetX Library: State management library for Flutter applications.
* Other Libraries and Tools: Additional libraries and tools as per project requirements.
* Language Used: Dart programming language for app logic and Flutter framework for UI development.

Hardware Requirements:

1. Desktop or Laptop Computer: A computer system capable of running development tools and compiling code efficiently.
2. Mobile Device (Android) for Testing: A physical device or emulator to test the application on real-world environments.
3. Web Camera (Optional): Required for testing camera functionalities if your application includes features that utilize the device's camera.
4. Reliable Internet Connection: Necessary for downloading dependencies, accessing documentation, and testing network-dependent features.
5. Adequate Storage Space: Sufficient storage space on your development machine to store project files, dependencies, and build artifacts.
6. Good Processing Power: A system with decent processing power to ensure smooth execution of development tasks, such as compiling code, running emulators, and testing applications.
7. RAM: Sufficient RAM to handle memory-intensive tasks and run development tools and emulators simultaneously.

6.Benefits



1-Empowered Independence: BeMyEyes fosters independence among visually impaired individuals by providing them with a tool to navigate and explore their environment autonomously. With the ability to identify objects and obstacles in real-time, users can confidently move through spaces without constant assistance, promoting a sense of self-reliance and empowerment.

2-Accessible Information: Through BeMyEyes, users have immediate access to valuable information about their surroundings. By leveraging object recognition technology, the app identifies and verbalizes objects and obstacles detected by the device's camera. This access to information enables users to make informed decisions and interact more effectively with their environment, enhancing their overall quality of life.

3-Heightened Safety and Awareness: One of the primary benefits of BeMyEyes is its contribution to user safety and awareness. By providing real-time feedback on detected objects and obstacles, the app helps users navigate potential hazards and avoid collisions. This heightened awareness of their surroundings not only reduces the risk of accidents but also instills a greater sense of confidence and security in users as they move through different environments.

Overall, BeMyEyes serves as a transformative tool for visually impaired individuals, empowering them to lead more independent, informed, and socially connected lives.

© 2022 Richard Szeliski, The University of Washington

Chapter 2 Object Recognition Overview



1.Introduction to object recognition

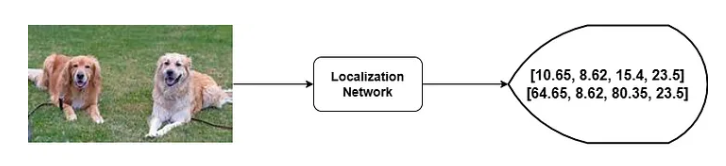
Object detection is a remarkable computer vision Technique which enables the computer to identify specific objects within an image and to also determine their precise location within the image.

Before object detection, Convolutional Neural networks provides prominent methods for classifying images into different classes. We can feed a picture into a Convolutional Neural Network and the network is able to identify if the picture is that of a dog, or a cat or a car etc. This task is called **Image Classification.**

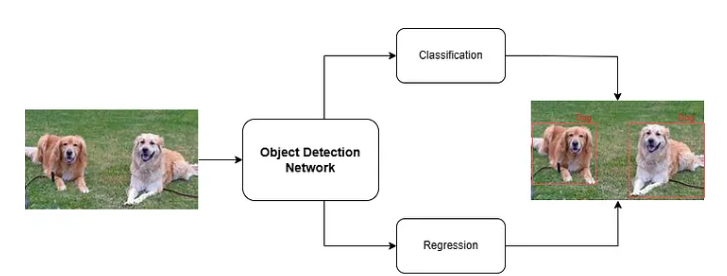
2.Object Localization

Object Localization refers to the task of precisely identifying and localizing objects of interest within an image. It plays a crucial role in computer vision applications, enabling tasks like object detection, tracking, and segmentation.

In this approach, the algorithm uses regression technique to determine the coordinates of the box’s top-left and bottom-right corners, effectively defining a rectangular region around the detected object.



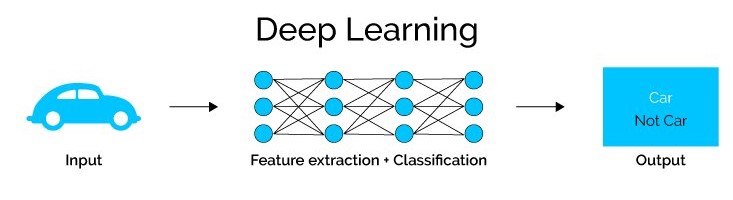
Object Detection is a combination of image classification and object localization. With these two techniques combined, an objected can be detected within a picture (classification) and also its position predicted through the use of bounding boxes (localization).



The object detection process typically involves two main components. First, it aims to identify and locate all the objects present in an image. Second, it entails predicting the bounding boxes that encapsulate each of these identified objects.

3.Deep Learning

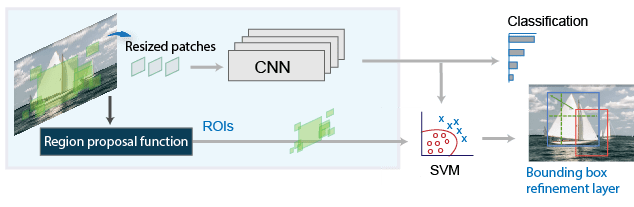
Deep learning is a subdomain of machine learning which takes on learning the representations from the data. It consists of successive hidden layers of representations and extracts more comprehensive and complex features as layers progress. The models on deep learning consist of neural networks containing the latent variables. Due to its supremacy over ever-increasing data, there has been a rise in the application of deep learning in various fields in recent years. They have been essentially used in applications such as but not limited to image recognition, NLP, medical data analysis, and bioinformatics. Specifically, due to the significant increase in textual data and records, there has been a development in traditional NLP methods to move toward utilizing the potential of deep learning. Techniques such as recurrent neural networks (RNNs) and long short-term memory networks have proven effective compared to previous methods. One of the objectives of deep learning is to convert unstructured data into structured data to make predictions more accurately when compared to other machine algorithms.



Abdul4code | Medium

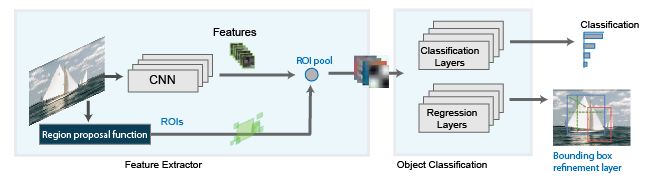
4. R-CNN

N detector first generates region proposals using an algorithm such as Edge Boxes. The proposal regions are cropped out of the image and resized. Then, the CNN classifies the cropped and resized regions. Finally, the region proposal bounding boxes are refined by a support vector machine (SVM) that is trained using CNN features.



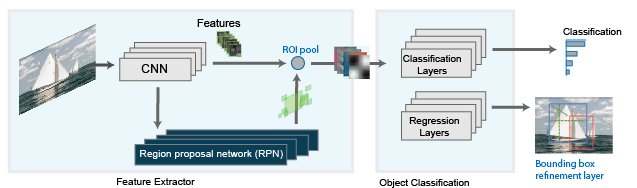
4.1 Fast R-CNN

As in the R-CNN detector , the Fast R-CNN[3] detector also uses an algorithm like Edge Boxes to generate region proposals. Unlike the R-CNN detector, which crops and resizes region proposals, the Fast R-CNN detector processes the entire image. Whereas an R-CNN detector must classify each region, Fast R-CNN pools CNN features corresponding to each region proposal. Fast R-CNN is more efficient than R-CNN, because in the Fast R-CNN detector, the computations for overlapping regions are shared.



4.2 Faster R-CNN

The Faster R-CNN[4] detector adds a region proposal network (RPN) to generate region proposals directly in the network instead of using an external algorithm like Edge Boxes. The RPN uses Anchor Boxes for Object Detection. Generating region proposals in the network is faster and better tuned to your data.



©1994-2024 The MathWorks, Inc.

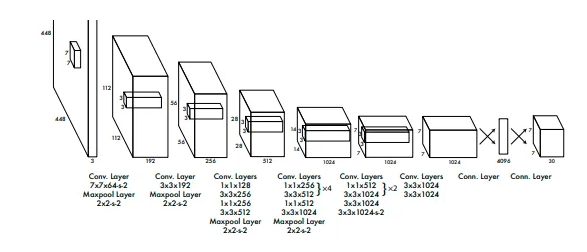
5. YOLO Algorithm (You Only Look Once)

The YOLO (You Only Look Once) algorithm revolutionized object detection by streamlining the process into a single pass over a convolutional neural network (CNN), eliminating the need for multiple stages of classification and region proposals. Instead of relying on region proposals, YOLO divides the input image into fixed-sized grids, typically 7x7 in the case of YOLO version 1. Each grid cell is responsible for predicting objects within its region, effectively replacing the need for region proposals.

Within each grid cell, YOLO performs both feature extraction and prediction tasks concurrently using a single neural network. This network processes the entire image and outputs predictions for object classes and bounding boxes directly. By reducing the object detection problem to a regression task, YOLO generates predictions for bounding box coordinates and object classes directly from the extracted features of each grid cell.

In essence, YOLO achieves object detection by transforming the input image into a set of grid cells, extracting features from these cells using a CNN, and making predictions for object classes and bounding boxes within each grid cell. This unified approach allows YOLO to achieve remarkable speed and efficiency in object detection tasks.

5.1 The YOLO Architecture



5.2 ANCHOR BOXES

In grid-based approaches like YOLO, each grid cell makes a single prediction, posing a challenge when multiple objects are present within a grid. To address this challenge, anchor boxes are introduced.

Anchor boxes are predetermined boxes with varying widths and heights, chosen based on the dataset's object characteristics. For example, in a dataset containing people and cars, cars typically have wider bounding boxes, while people have taller ones. Hence, two anchor boxes—one wider and one taller—are defined to accommodate both object types within each grid.

By incorporating anchor boxes, each grid can now detect multiple object types simultaneously. For instance, using two anchor boxes per grid enables the model to potentially detect up to 98 objects (49 grids x 2 anchor boxes).

The selection of suitable anchor boxes involves methods like clustering. Through clustering algorithms, width and height values of bounding boxes in the dataset are grouped into clusters, each representing object categories like people or cars. The centroid of each cluster serves as the anchor box for its corresponding category, facilitating accurate object detection.

6. MobileNet

Mobilenet is a light-weight computer vision model designed to be used in mobile applications. It is a is a computer vision model open-sourced by Google and designed for training classifiers. It uses depthwise convolutions to significantly reduce the number of parameters compared to other networks, resulting in a lightweight deep neural network. MobileNet is Tensorflow’s first mobile computer vision model. It uses depthwise separable convolutions to significantly reduce the number of parameters compared to other networks with regular convolutions and the same depth in the nets. This results in lightweight deep neural networks.

A depthwise separable convolution is made from two operations:

1. Depthwise convolution
2. Pointwise convolution

MobileNet is a class of convolutional neural network (CNN) that was open-sourced by Google, and therefore, provides an excellent starting point for training classifiers that are insanely small and insanely fast.

The speed and power consumption of the network is proportional to the number of multiply-accumulates (MACs) which is a measure of the number of fused multiplication and addition operations.

6.1 MobileNet Depthwise Separable Convolution

This convolution originated from the idea that a filter’s depth and spatial dimension can be separated, thus, the name separable. Let’s take the example of the Sobel filter used in image processing to detect edges.

You can separate the height and width dimensions of these filters. Gx filter can be viewed as a matrix product of [1 2 1] transpose with [-1 0 1].

You’ll notice that the filter has disguised itself. It shows it had nine parameters, but it has six. This is possible because of the separation of its height and width dimensions.

The same idea applies to a separate depth dimension from horizontal (width\*height), which gives us a depthwise separable convolution where we perform depthwise convolution. After that, we use a 1\*1 filter to cover the depth dimension.

One thing to notice is how much the parameters are reduced by this convolution to output the same number of channels. To produce one channel, we’d need 3\*3\*3 parameters to perform depth-wise convolution and 1\*3 parameters to perform further convolution in-depth dimension.

But if we’d need three output channels, we’d only need 31\*3 depth filter, giving us a total of 36 ( = 27 +9) parameters, while for the same number of output channels in regular convolution, we’d need 33\*3\*3 filters giving us a total of 81 parameters.

6.2 Advantages of MobileNet

MobileNets are a family of mobile-first computer vision models for TensorFlow, designed to effectively maximize accuracy while being mindful of the restricted resources for an on-device or embedded application.

MobileNets are small, low-latency, low-power models parameterized to meet the resource constraints of a variety of use-cases. They can be built upon for classification, detection, embeddings and segmentation.

© 2022 Richard Szeliski, The University of Washington

7. Conclusion

In object recognition, advanced techniques like YOLO and RCNN utilize innovative approaches such as grid-based predictions and anchor boxes to efficiently detect objects in images. These methods leverage deep learning architectures and carefully annotated datasets to achieve accurate and robust object detection. Additionally, techniques like separable convolutions optimize computational efficiency by reducing the number of parameters required for convolutional operations, further enhancing the performance of object recognition systems.

Chapter 3 Implementation and Functionality Overview

1. Introduction

1.1 Implementation Process

Implementing a mobile application for blind people entails a multifaceted process that involves transforming conceptual ideas into tangible software solutions. This introductory section provides an overview of the implementation journey, highlighting key steps and considerations.

1.2 Technologies and Tools

The development of the application relies on a set of technologies and tools selected to facilitate efficient implementation and robust functionality:

Flutter Framework: The application is built using the Flutter framework, providing a comprehensive set of widgets and libraries for cross-platform mobile development.

Dart Programming Language: Dart serves as the primary programming language for implementing application logic and user interface elements. Its simplicity and readability contribute to streamlined development and maintenance of the codebase.

TensorFlow Lite: TensorFlow Lite integration enables object recognition functionality within the application. Leveraging pre-trained machine learning models, TensorFlow Lite facilitates real-time detection and identification of objects captured by the device's camera.

Get Package: The Get package is utilized for state management within the application. Employing reactive programming principles, Get simplifies state management and facilitates seamless communication between different application components.

Permission Handler: The Permission Handler plugin manages platform permissions, particularly camera access permissions required for object recognition. It ensures compliance with privacy and security standards while providing essential features to users.

Flutter TTS Plugin: Integration of the Flutter TTS plugin enables text-to-speech conversion within the application. This feature enhances accessibility and usability for visually impaired individuals by providing auditory feedback.

2. User Experience Design

Accessible User Interface: The user interface (UI) elements, such as buttons, text labels, and navigation, are designed with accessibility in mind. This includes using high-contrast colors, clear and concise text, and intuitive layouts.

Focus on Voice and Text Output: Given the target audience's reliance on auditory feedback, the design emphasizes voice and text output features. The integration of the Flutter TTS plugin enables text-to-speech conversion, ensuring that information is conveyed audibly to users.

Feedback Mechanisms: The application provides auditory feedback to users to confirm actions and provide context. For example, when users interact with UI elements or initiate actions, such as tapping buttons, they receive spoken feedback through the Flutter TTS plugin, enhancing the overall user experience.

3. Enhancing Flutter App Quality with Widgets

3.1 Optimizing Code Quality

* Consistent Use of const Constructors: Throughout the application, const constructors are employed where appropriate. For instance, in the MyApp class, the use of const MyApp() ensures that instances of the class are immutable and promotes efficient memory usage.
* Effective Variable Management: The judicious use of final and late keywords ensures proper variable immutability and initialization. For instance, variables like cameras and detectedLabels are appropriately declared as late to defer their initialization until runtime.
* Robust Exception Handling: The code demonstrates robust exception handling, particularly evident in the initCamera method, where permission denial is gracefully managed. By handling potential runtime errors, the application maintains stability and enhances user experience.
* Logging for Debugging: Utilizing the dart:developer library, the code incorporates logging mechanisms to aid in debugging and understanding program flow. Logging statements, such as in the initCamera method, provide valuable insights during development and troubleshooting phases.

3.2 Harnessing Widgets

* Modular Widget Composition: Various Flutter widgets, including MaterialApp, Scaffold, Column, Text, ElevatedButton, and RichText, are strategically utilized to compose the user interface. These widgets facilitate the creation of visually appealing and interactive UI components.
* Reusable Component Encapsulation: Reusable components, such as the StartPage and CameraView, encapsulate UI elements and associated functionality. This modular approach promotes code reusability, simplifies maintenance, and enhances readability.
* State Management with GetX: The CameraView widget employs the GetBuilder widget from the GetX state management library. This implementation efficiently manages UI state updates, ensuring responsiveness and scalability in the application.

