A secure mobile application for speech to text conversion using artificial intelligence techniques

Kartik Bhasin [1], Arjun Goel [1], Gaurav Gupta [1], Sanjay Kumar Singh [1] and Abhijit Bhowmick [1]

[1] *Vellore Institute of Technology, Tamilnadu, India*

Abstract

Optical Character Recognition (OCR) of papers has tremendous practical value given the prevalence of handwritten documents in human exchanges. Optical character recognition is an AI-enabled algorithm that makes it possible to convert many kinds of texts or photos into editable, searchable, and analyzable data. It requires high computation resources making it difficult to run without proper setup. But with the advent of cloud computing, it is now very much possible to bring this to handheld devices as well. In this paper we discuss ways that enable the user to use optical character recognition as well as object detection from their mobile phone itself. Voice recognition or speech to text is a very convenient way of editing or writing documents. The voice signal has to be recognized and processed alongside noise to get the desired result. In this paper, we are going to discuss an app made by us that can get the pdf of scanned text as well as support speech-to -text features and get its pdf as well.

Keywords

OCR, AI, Cloud-Computing, voice-recognition, speech-to-text, pdf.

1. Introduction

In a time where everything is handled through internet computers and servers, handwritten text or even text through books does not catch up. Neither would you be able to share them with someone far away or multiple people nor would you be able to carry them with you everywhere. You sometimes might forget your book or the conference you are going to does not allow you to bring anything with you. So what can be the solution?

We can save the books word by word on the internet or in this case our phone. We do this usually in text or pdf format. But why though? Why not just take a photo of the page you want to carry? Well, a single photo will take at least 4MB of size in today's day and age. Not only that there is also a chance of the photo taken being blurry or the text becoming unable to read due to the saturation of the image.

In this project, we have used OCR to scan the image and extract the text from it. We then convert it into a pdf and give it back to the user for use. PDF files are much less in size compared to their image counterparts and much easier to read as well. With PDF there also comes the option of increasing the font size or changing the font style according to the need of the user. We have taken help for this from Chinnu Muniyappan's article on text extraction from PDF. [1]

WINS 2023: Workshop on Intelligent Systems, May 20 – 21, 2023, Chennai, India.
EMAIL: sanjaykumar.singh@vit.ac.in (Sanjay Kumar Singh)
ORCID: 0000-0002-5760-1865 (Sanjay Kumar Singh)

Optical Character Recognition (OCR) is a logic that converts the input text or the text extracted from a document into the machine-level form. Today, OCR not only helps digitize handwritten medieval manuscripts, it also helps convert typed documents into digital form. This makes it easier to find the information you need without having to dig through piles of documents and files to find what you need. Organizations meet digital preservation needs such as historical data, legal documents, and educational permanence.

OCR systems mainly rely on feature extraction and discrimination/classification of these features (based on patterns). Handwritten OCR is gaining attention as a subset of OCR. Based on the input data, it is further classified into offline and online systems. Offline systems are static systems where the input data is in the form of scanned images, whereas online systems' type of input is more dynamic, allowing the pen Based on the movement ahead. Online systems are therefore considered more complex and advanced as they solve the problem of duplication of input data that exists in offline systems. We will be using an online system in the paper.We also have implemented the algorithm of speech to text as well. Speech-to-Text is applied to create transcripts, captions, or any other kind of text that businesses need today. It works by "translating" the speech into a spelled out form word by word.It is also a simple way to promote independence for students with disabilities. Students can write whatever comes to mind at any time using intelligent speech-to-text technology. This also allows them to complete homework without parental assistance.

These digital versions are of great benefit to children and young people with reading difficulties. For this reason, digital text can be used in some software packages to improve readability. The text is copied or read using techniques such as optical scanners or special circuit boards, and software processes further analysis. The primary use of OCR is converting printed legal or historical documents to PDF. Users can save their papers in PDF format, and then modify, format and analyze them as if they were written in a word processor.

2. Motivation

2.1. Optical Character Recognition

2.2.1. Image Pre-Processing

In the first stage, the technology will convert the document from its physical shape into a picture that can be understood by the machine. The purpose of this phase is to remove unwanted aberrations while ensuring that the representation of the machine is accurate. This is done through edge and contour detection. We have referred to Rupak Kargi’s article to learn more about object detection in flutter interface [2]. The line connecting all the points along an image's edge that have the same intensity is referred to as the contour. Contours are useful for object detection, determining the size of an object of interest, and shape analysis. We will basically take the largest contour available to us in the picture and then the edges of the contour are marked to separate it from the rest of the image.

We are taking the largest contour available because the largest contour in the image would obviously be the text written either on paper or on some other material on which we have to apply our logic and process further. The concept is subsequently transformed to a black and white rendition, evaluated for bright vs. dark regions (characters). Then use an OCR system to split the image into individual parts such as spreadsheets, text, and graphics.

2.2.2. AI Character Recognition

Suppose life was so simple that there was only one letter in the alphabet, A. Even then, one can observe that the use of OCR would be a very difficult problem because not every person writes the letter A in a similar manner. This issue is persistent even with printed text, because books and other digital documents are written and printed in many different styles (fonts). Even taking an example of a single letter A, it can be seen that it is printed in many different forms. See fig 1.

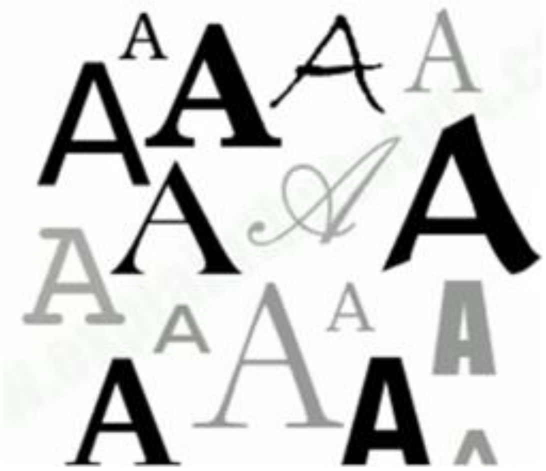


Figure 1: See that there are some differences between all capital A written in different styles and printed in different types of fonts, but there are also fundamental similarities between them. One can see that most things consist of two diagonal lines that meet at the top center, with a horizontal line in the middle.

There are actually two ways to solve this problem. One is that it can either recognize whole characters, that is, pattern recognition. While other is to recognize and identify the individual lines and dashes that make up a character (feature recognition). We have taken help from Chris Woodfords article on OCR [3]. Let's look at these in order.

2.2.3. Pattern Recognition

We could standardize a certain font or text style for OCR. OCR-A, a special kind of font was developed in the 1960s and was used for things like bank checks. Each character is exactly similar in width (Example of a monospaced font) and the stroke is carefully designed so that each character could be easily distinguished from all the others. Check printers are designed to use all of these fonts, and OCR systems are also designed to recognize them. By normalizing these simple fonts, the OCR problem has become relatively easy to tackle. But the problem is that most things printed in the world are not written in this special font, OCR-A. And nobody uses this font for handwriting!

2.2.4. Feature Detection

This is also called intelligent character recognition (ICR), it's a much more complicated way to detect characters. The question now is how can we differentiate between letters and characters all in different fonts? We can actually set rules for a particular character or letter. For example, if two oblique lines meet at a point at the top, center, and there is a horizontal line between them at a distance apart, then that letter should be recognized as letter A. This would ensure that the letter A is recognized no matter the font. So, instead of recognizing the character as a whole sample, we try to detect its individual components from which the character is derived or its characteristics. For example, here is a word: a rat in fig 2.

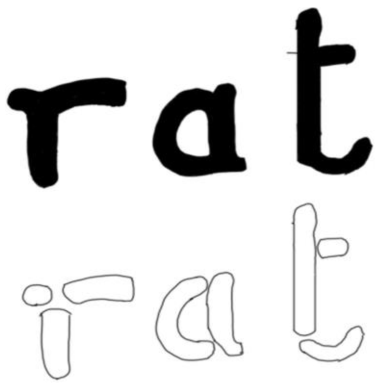


Figure 2: Feature detection of word rat

2.2.5. Post-Processing

After the end of feature and pattern recognition there's a chance that some of the letters may be misclassified and the spelling of words may be confused. To address this issue, we compare the retrieved words to the word list and correct any errors. And finally, all the parts come together. Characters, words, and sentences are combined to form the final output document. The AI tries to correct all the errors in the final file during the Post-Processing phase. One approach would be to teach the AI algorithm a glossary of terms that would be present in the document, The output of the AI is then limited to these words or formats to verify that no interpretation is outside of the vocabulary.[4] There have also been several alignment methods that have been developed to align the outputs of different OCR engines to improve the efficiency of the character recognition. Also, refer to fig 3 for the final flowchart of the OCR.

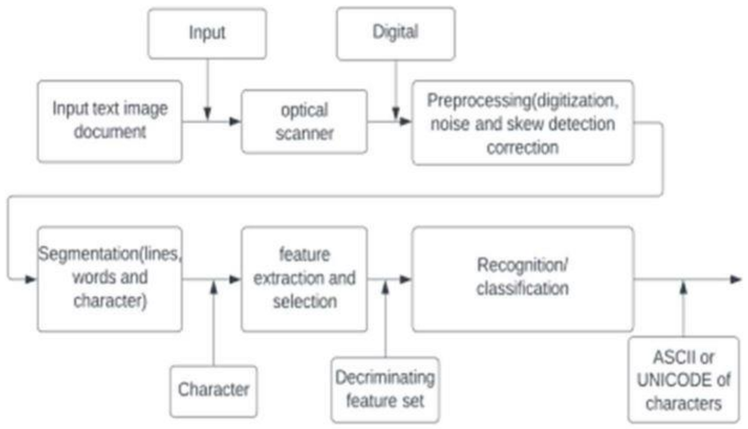

Input
Digital
Input text image document
optical scanner
Preprocessing(digitization, noise and skew detection correction
Segmentation(lines, words and character)
feature extraction and selection
Recognition/ classification
Character
Decriminating feature set
ASCII or UNICODE of characters

Figure 3: Flowchart for working of OCR

2.2. Speech Recognition

Speech to text works by catching an audio signal and delivering its verbatim transcriptions that can be edited on a given device. This is done by the software using voice recognition. A trained model, based on linguistic algorithms that sorts auditory signals from words spoken by a person and converts those signals into text using characters known as Unicode. Speech-to-text conversion is performed using a complex machine learning model with multiple steps. [5] Let's see step by step how this works:

- When sounds come out of someone's mouth to create words, it also makes a series of vibrations

- Then, using an analogue to digital converter, speech-to-text technology picks up these vibrations and converts them into a digital language.
- The analog-to-digital converter takes sounds from an audio file, measures the waves in great detail, and filters them to distinguish the relevant sounds.
- The sound is then divided into hundredths or thousandths of a second.
- They are then matched to phonemes. A phoneme is a unit of sound.
- This ensures that any word from any other languages can be distinguished from each other
- For instance, the English language contains about 40 phonemes.
- A mathematical model that matches the phonemes to well-known sentences, words, and phrases then runs them through a network.
- Then, based on the most likely rendition of the audio, the text is displayed as text or as a computer based demand.

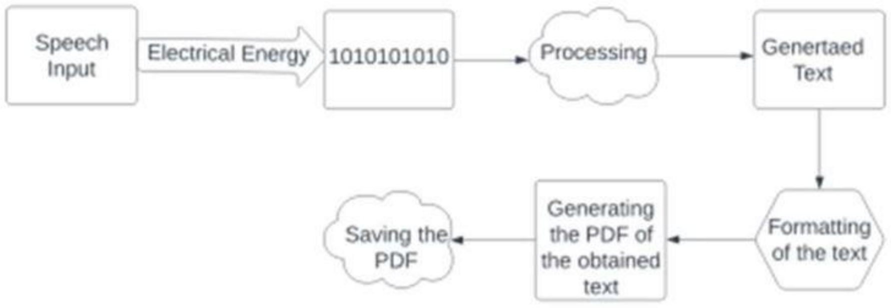


Figure 4: Flowchart for working of speech recognition.

The recorded user's voice that is sent for processing to our database might contain some noise either in static or dynamic form. It could be both physical or electrical noise, Whatever kind it may be it must be filtered out from the recorded voice of the user before being used to extract the text from it, There have been lots of research regarding the best filter for noise suppression in a voice, We would design a Neuro - Fuzzy System and use it as our filter for noise suppression. [6]

2.3. Flutter

Flutter is an open-source cross-platform toolkit used for building apps for mobile OS, web and desktop with the same code base. That means developers write code once and apply it to all the Platforms - IOS, Android or Web. It is created by Google and is one of the used tools in application development in the industry as described by Theodoros in his article. [7]

It combines the quality of native apps with the flexibility of cross-platform development. Flutter renders the same look as a native app: it draws the UI from scratch, not acting as a wrapper on top of native UI components as other frameworks do. Also it is an open source framework which means one can share their work on GitHub.

Flutter is not just a framework, it's a complete SDK for building apps. That means Flutter contains everything needed to create a user interface (UI), including the widgets for material design. Flutter uses Dart language which applies to both server and client-side development. We also have opted for flutter for developing our application and, to make it secure, is fast and has a user friendly-UI as well.

3. System Design

The system will consist of a series of various processes which will be processed from the selection of an image to converting into text or converting the speech signal to text.

3.1. Voice to text

When the user enters the app after signing in or registering themselves, they will see be given the option to go for text to speech or image to text. To use Speech to text, the user will have to press the microphone button once and start speaking. After getting done with the speech, users can either convert the text generated into PDF or simply copy the text and use it as per their requirements.

The voice file is filtered with the neural fuzzy system to decrease the noise and is sent to the database to get the text of the voice file.

3.2. Image Selection

Users can choose to select an image from the gallery or capture an image using the device's built-in mobile camera. Users can select images in both his commonly used JPEG and PNG image formats. When the user selects an image from the gallery, the mobile application accesses the image from internal storage. When the user is ready to take a picture, the application takes a picture with the built- in mobile camera and saves it in temporary storage. Photos are automatically deleted when the user exits the application. This also ensures that no storage space is wasted on unnecessary images.

When the user selects an image from the gallery or camera, the user is taken to the image preview page. This page displays a zoomable image, so the user double-checks the image to see if the correct image was uploaded by the user. Here the user can choose to press the "Tick" button to go to the next page or press the "Cross" button to select the image again. The image is then processed to get the biggest contour present in the image. The idea is to separate unnecessary objects present in the image and just focus on the area where the text to be extracted from the image is present. This decreases the computation power from the server side (OCR engine) and thus saves both time and resources.[8]

3.3. OCR Model

The contour detected image would then be sent over to our cloud server where we have the OCR algorithm which will extract the text from the image as already discussed. The extracted text is then returned to our application in JSON format. This is an asynchronous function that we have to explicitly define while writing the code for and await for the data. We then unpack the data and extract the text from it which is presented to the user in a text format. The user is again free to use the text directly or get the PDF of the text by pressing the 'GET PDF' button which then processes the text and converts it into a PDF file.

3.4. Application Interface

The virtual machine (VM) that Flutter apps run in enables stateful quick reloading of changes without the need for a full recompile while in development. The framework is free source with a flexible BSD license and has a strong ecosystem of third-party packages that supplement the core features. [9]

An abstraction for handling layout is provided by the rendering layer. We can create a tree of renderable items with this layer. These objects can be dynamically changed, and the tree will update its layout to reflect the modifications. The widgets layer is an abstraction of composition. There is a class in the widgets layer for each render object in the rendering layer. Additionally, we can define reusable class combinations in the widgets layer. The reactive programming model is introduced at this tier to make the app responsive. The color box, hue slider, and radio buttons are just a few of the places where the state can be altered. Changes must appear everywhere else when the user interacts with the UI. Even worse, if caution is not used, a little modification to one area of the user interface may affect other seemingly unrelated sections of code. Flutter and other reactive frameworks handle this problem differently by explicitly isolating the user interface from its underlying state. You only need to build a user interface (UI) description with React-style APIs, and the framework will use it to generate and/or update the user interface as needed.

Flutter widgets are modeled after immutable classes that resemble React components. the construction of an object tree. These widgets are used to manage an independent tree of objects for compositing as well as an independent tree of objects for layout. Flutter's primary function is to rapidly cycle through updated tree sections, demote entries to lower-level objects, and propagate tree change. The number of design concepts is kept to a minimum while still preserving a wide vocabulary. For instance, Flutter uses the same fundamental idea (a Widget) in the widgets layer to represent drawing to the screen, layout (positioning and scaling), user interaction, state management, theming, animations, and navigation. The concepts of Animations and Tweens occupy the majority of the design space in the animation layer. Layout, painting, hit testing, and accessibility are all described at the rendering layer using render objects. There are a large number of widgets, render objects, animation and tween types, and each of these circumstances leads to a large vocabulary for the proper terminology.

To maximize the number of options, the class hierarchy is purposefully short and broad, focusing on small, assemblable widgets that each do one thing exceptionally well. Even fundamental features like padding and alignment are implemented as independent components rather than being incorporated into the core since core features are abstract. (This contrasts with more conventional APIs, which incorporate elements like padding into the fundamental structure of each layout component.) To center a widget, for instance, you would wrap it in a Center widget rather than changing a notional Align attribute. Padding, alignment, rows, columns, and grids are all covered by widgets. These layout widgets lack a unique visual representation. Instead, their main function is to regulate a certain feature of the layout of another widget. Utility widgets in Flutter also benefit from this compositional strategy.

For instance, the often-used widget Container is composed of various widgets that are in charge of the layout, painting, positioning, and scaling. Reading the source code of Container reveals that it is composed of the DecoratedBox MaterialBox, Rows, Columns, Padding, Align, Sateful and Stateless widgets. Being able to study any widget's source code irrespective of the position of the widget in the tree, is one of Flutter's distinguishing features. One can simply compose it to achieve a unique effect rather than the usual subclassing of the Container. Integrate existing widgets in fresh ways, or just make a new widget by drawing ideas from MaterialContainer.

4. Implementation

Using the system design described above we have made an app using the flutter platform that integrates all the components that we have discussed so far. We have created this app keeping the convenience and ease of use of the user in our mind. We have developed the application as follows:

- Create a flutter application project in an IDE (We have used Virtual Studio Code) by running flutter to create an ‘App name’ and get the basic template for the app.
- Create a material app that would return a scaffold widget in the homepage. Set routes of different pages that would be called in the app. By name they would be the login page, homepage, picture choosing page and the output page. (The output page would be shared between both the OCR and speech to text processes.)
- Design the login page for authentication and connect this page to the firebase authentication to make the app secured. This would also keep the record number of registered users in the cloud database which would then help us to decide if we have to scale our app to provide for more number of users.
- After this, we design the homepage by using widgets on top of the scaffold and providing buttons for both voice access and camera access. A button for straight away copying the recognized text from the speech has also been made on the top right corner.
- The microphone button is then connected to the voice recognition and speech to text algorithm and also with the firebase to store the data. The recorded voice is sent to the cloud services where computations will happen and the text is returned.

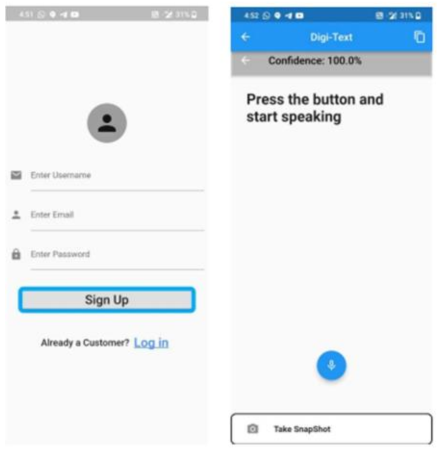


Figure 5: Login Page and the home Page of the app

- Make a list view widget to display the text on the screen of the phone in a scrollable format. Also, define buttons that allow users to get the PDF of the text. The back button should also be routed to take the user back to the login screen on being pressed.

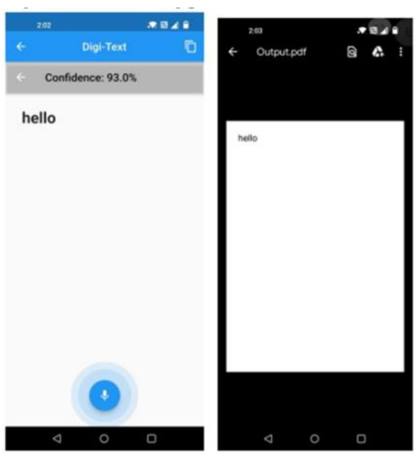
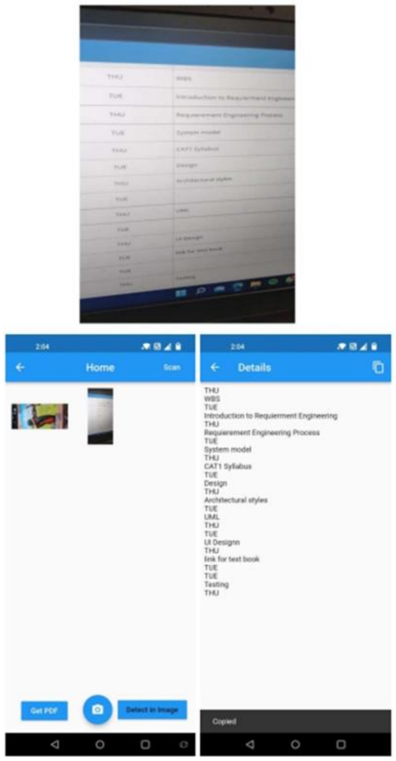


Figure 6: Voice recognition and PDF of the output

- Use the image picker module to define the on-press function of the image button to let the user choose between the camera and the gallery.



```
THU
WBS
TUE
Introduction to Requierment Engineering
THU
Requierement Engineering Process
TUE
System model
THU
CAT1 Syllabus
TUE
Design
THU
Architectural styles
TUE
UML
THU
TUE
UI Designn
THU
link for text book
TUE
TUE
Testing
THU
```

Figure 7: Image to Text and selection procedure

- Once the image has been selected the snapshot or the reference of the image is saved and the user is moved to the next screen. Since this could take a while, we will define the data that is returned from the API as asynchronous for which the application will wait. A circular progress bar would be displayed during this period for indication to the user that their data is being processed and generated.
- The data is processed and the output would then be received by the app and displayed to the user with the option to convert it to pdf or get a copy of the text. The pdf would be saved on the user's device. (We can also automatically save the PDF on the users logged in google account drive.)

5. Conclusion and Future Work

In this paper, a mobile application has been proposed that is secured with authentication and allows the user to get text from their voice and from images as well. Images can be either taken from the phone camera or the gallery of the phone. The final app has been tested and the results obtained are promising.

As part of future work since we are already getting the text in the Unicode format, we can insert an option for a translator that could translate the scanned text into any wanted or preferred language. We could also add an object detection algorithm that could recognize the object that is being kept in front of the camera as we have already implemented the algorithm for finding the largest contour or the object in focus for extraction of text.

6. Acknowledgements

We would like to thank our university for providing and supporting us with the necessary resources to write this paper. We would also like to show our appreciation for websites like StackOverflow and geeks for geeks which helped us to debug our code whenever we got stuck.

7. References

[1] Chinnu Muniyappan,"5 Ways to Extract Text from PDF document",Available:https://www.syncfusion.com/blogs/post/ 5-ways-to-extract-text-from-pdf-documents-in-flutter.aspx#ex tract-all-the-text-from-a-pdf-document

[2] Rupak Karki, (2020), "Detecting Objects in Flutter" Available:https://towardsdatascience.com/detecting-objects-in -flutter-4fe8cfccef14

[3] Chris Woodford,(2021) "Optical character recognition (OCR)". Available:https://www.explainthatstuff.com/how-ocr-works.ht ml

[4] Che Abdul Rahman, Anis Nadiah & Ho Abdullah, Imran & Zainuddin, Intan & Jaludin, Azhar. (2019). THE COMPARISONS OF OCR TOOLS. MALAYSIAN JOURNAL OF COMPUTING. 4. 335. 10.24191/mjoc.v4i2.5626.

[5] Kothadiya, Deep & Pise, Nitin & Bedekar, Mangesh. (2020). Different Methods Review for Speech to Text and Text to Speech Conversion. International Journal of Computer Applications. 175. 9-12. 10.5120/ijca2020920727.

[6] J. Kumar, G. Parmar, N. Gupta and R. Kapoor, "Environmental Noise Cancellation by Using Neuro Fuzzy Adaptive Filtering," 2015 Fifth International Conference on Communication Systems and Network Technologies, 2015, pp. 1157-1162, doi: 10.1109/CSNT.2015.114.

[7] Theodoros 'Theo' Karasavvas ,(2021),"Why Flutter is the most popular cross-platform mobile SDK". Available:https://stackoverflow.blog/2022/02/21/why-flutter-i s-the-most-popular-cross-platform-mobile-sdk/

[8] "The official package repository for dart and flutter" supported by Google, Available:https://pub.dev/

[9] Anmol Gupta,(2021),"Live object detection App with flutter andTensorFlow".Available:https://medium.flutterdevs.com/liv e-object-detection-app-with-flutter-and-tensorflow-lite-a6e7f7 af3b07