

ARTIFICIAL INTELLIGENCE-POWERED MOBILE APPLICATION TO HELP VISUALLY IMPAIRED PEOPLE

Submitted in partial fulfillment of the requirements for
the award of
Bachelor of Engineering degree in Computer Science and Engineering
by

Harsh Bardhan Mishra (38110196)

Hrituraj Sutradhar (38110202)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade "A" by NAA

JEPPIAAR NAGAR, RAJIV GANDHI SALAI,

CHENNAI - 600 119



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Harsh Bardhan Mishra (38110196)**, **Hrituraj Sutradhar (38110202)** who carried out the project entitled as **Artificial Intelligence-powered Mobile Application to help visually impaired people** under my supervision from November 2021 to April 2022.

Internal Guide

Dr.L.Lakshmanan, M.E., (Ph.D.,)

Head of the Department

Dr. S. VIGNESHWARI, M.E., Ph.D.,

Dr. L. LAKSHMANAN, M.E., Ph.D.,

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I **Harsh Bardhan Mishra (38110196)**, **Hrituraj Sutradhar (38110202)** hereby declare that the Project Report entitled “**Artificial Intelligence-powered Mobile Application to help visually impaired people**” done by us under the guidance of **Dr. L.Lakshmanan, M.E., (Ph.D.,)** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE: 21.04.2022

SIGNATURE OF THE CANDIDATE: Harsh Mishra

Hrituraj Sutradhar

PLACE: Chennai

ACKNOWLEDGEMENT

I am pleased to acknowledge our sincere thanks to the Board **of Management** of **Sathyabama Institute of Science and Technology** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala, M.E., Ph.D., Dean**, School of Computing, **Dr. S. Vigneshwari, M.E., Ph.D.**, and **Dr. L. Lakshmanan, M.E., Ph.D., Heads** of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Mr. Vignesh. R, M.E., (Ph.D.,)** for his valuable guidance, suggestions and constant encouragement paved the way for the successful completion of our project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

With the rise of development around software engineering, humanity hasn't implemented enhancements that we can use for visually impaired people. With the development of artificial intelligence and machine learning techniques, engineers can use to infuse "intelligence" even in dumb computers and with the ease of accessibility, we can extend this "intelligence" to our smartphones to help the visually impaired people understand their surroundings and receive a helping hand during their day-to-day activities.

Our mobile application bridges the gap between visually impaired people and the visual world by leveraging the power of Deep Learning which can be made accessible even on low-end devices with a lucid User-Interface that allows them to better understand the world around them.

Our primary purpose is to leverage and study how Deep Learning architecture and easy prototyping tools can help us develop applications that we can quickly render, even on low-end devices. With this application, we aim to have a one-stop solution to allow blind or partially blind People to understand their surroundings better and cope with the dynamic world ahead of them.

Our mobile application allows the Users to leverage Image Captioning Architecture to generate real-time insight into their surroundings while using Natural Language Processing to speak out lucidly. The cornerstone of our mobile application is its User Interface which would infuse a lucid experience for the User with its ease of handling and use.

TABLE OF CONTENT

CHAPTER NO.	TITLE	PAGE NO.
1	INTRODUCTION	1
1.1	MOTIVATION	1
1.2	IDE	2
1.3	OBJECTIVES	3
1.4	OVERVIEW OF THE PROJECT	4
2	LITERATURE SURVEY	5
3	SCOPE AND REQUIREMENT ANALYSIS OF PRESENT INVESTIGATION	7
3.1	EXISTING SYSTEM	7
3.2	PROPOSED SYSTEM	7
3.3	TECHNOLOGY USED	8
3.4	REQUIREMENT ANALYSIS	8
4	METHODOLOGY AND ALGORITHM USED	10
4.1	INTRODUCTION TO MACHINE LEARNING	10
4.2	INTRODUCTION TO DEEP LEARNING AND TRANSFER LEARNING	11
4.3	TECHNOLOGIES USED	14
4.4	IMAGE CAPTIONING MODEL	20
4.5	OBJECT DETECTION MODEL	24
4.6	SYSTEM ARCHITECTURE	27
5	RESULTS AND DISCUSSION	30
6	CONCLUSION AND FUTURE WORK	34
	REFERENCES	35
	APPENDIX	37
	PLAGIARISM REPORT	100

LIST OF FIGURES

FIGURE NO	TITLE	PAGE NO
1	Explaining Convolutional Neural Network	12
2	Explaining Transfer Learning	13
3	Flickr8k Dataset	20
4	Recurrent Neural Networks and Convolutional Neural Networks	21
5	Image Caption generation using CNN and RNN	23
6	Image Caption generation in action	24
7	Object detection in Tensorflow	25
8	Integration of Tensorflow Lite	26
9	Tensorflow Object detection in action	27
10	System architecture diagram	29
11	Detecting objects on mobile application	31
5,7	Detecting currency notes on the application	32
5.8	Detecting text from images on the application	33

LIST OF ABBREVIATIONS

ABBREVIATION	EXPANSION
ML	Machine Learning
IDE	Integrated Development Environment
UI	User Interface
API	Application Programming Interface
NLP	Natural Language Processing
CNN	Convolutional Neural Network
UX	User Experience

CHAPTER 1

INTRODUCTION

Vision is one of the most important human senses and helps us understand the perception of the environment around us. This information is not available through conventional methods for visually impaired people, and they have to depend on external interventions to utilize it. However, it leads to a lot of problems in their safety and mobility, with the visually impaired people having to rely on external factors for everyday activities. The rise of artificial intelligence and mobile application development has fused the user experience to provide the best possible user experience for the general mass. Applications like Tik Tok, Instagram, Amazon, and Spotify use real-time machine learning algorithms to accelerate their data-driven insights and bring the best possible user experience. However, not much research and implementation have been done by researchers and engineers to help visually impaired people understand their visual scenes. Existing work has been mainly focused on helping understand the components and characteristics of the environment, which is not fully scalable. This project aims to propose a solution for visually impaired people to better understand their environment, with specific features implemented to allow better them to navigate and interact with the environment.

1.1 MOTIVATION

At present, the development of assistive technology is still in its infancy. Assistive technology (AT) is helpful for people who are striving to maximize their efficiency and work output but are limited by their disabilities. Assistive technologies can help people with specific disabilities carry out their tasks much better, reduce the social and emotional burden and help them contribute to the professional outlook of their career.

Assistive technology for visually impaired people saw initial traction in the 2010s, developing watches, yardsticks, special glasses, and other devices. However, the real boost came in the mid-2010s when mobile application development boomed. The mobile application development boom was fueled by the rise of the Android and iOS platforms. The capability to embed high-functioning applications in the user's devices was deemed proper to bring this technology to light.

The development of machine learning frameworks focused on mobile applications, like Tensorflow Lite, also helped herald the integration of assistive technology into the mobile application development process. However, the hype died down very slowly even though mobile application development has continued to grow. It has been because of a lack of a firm business plan, lack of a technical roadmap, and lack of specific talent to build a team.

It has introduced a lot of half-baked applications in the market since the development of assistive technology has been a slow and inefficient process. While picking up this problem, we wanted to focus on the visually impaired population specifically. Today, India is home to around 12 million visually impaired people, one-third of the world's total visually impaired population. To prevent the visually impaired population from being left behind, we decided to focus on building a mobile application with modern machine learning and deep learning techniques and frameworks.

1.2 IDE

Visual Studio Code (VSC) is a desktop-based code editor developed by Microsoft. It is used to create and edit code and debug programs. We used VS Code for developing the mobile application alongside an Android 11-based smartphone as an emulator for development, testing, and debugging. Jupyter Notebooks were used for machine-learning modeling and creating the deployment models. The

environment was managed using Anaconda, an open-source distribution of Python and R, to perform data science and machine learning.

1.3 OBJECTIVES

The main aim of this project was to develop a mobile application that would capitalize on machine learning models developed by us to help visually impaired people to navigate and understand their surroundings much better. The most fundamental challenge was integrating our machine learning models while ensuring a smooth user experience and keeping the application size small and flexible. The project provides multiple features like:

- All machine-learning-powered features, except image captioning, work wholly offline and do not require any internet connection.
- All offline features work in real-time and don't need any processing time for models to make predictions.
- Every feature, including image labeling to currency detection, uses text-to-speech to give the application user a more natural voice.
- The application features an intuitive user experience with a minimum number of buttons, each having a large size.

1.4 OVERVIEW OF THE PROJECT

Chapter 1, deals with the introduction to the project where the existing system is being discussed.

Chapter 2, a detailed description of the literature survey of the papers which are referred during the course of the project was summarized.

Chapter 3, gives a brief explanation on the aim and scope of the project. Here the proposed system has been compared with the existing system. The software and

hardware requirements are provided along the system design, architecture overall project.

Chapter 4, the Results and Discussion along with the screenshots of each module has been depicted.

Chapter 5, deals with the summary and conclusion of the project. It also includes the future scope of the project.

CHAPTER 2

LITERATURE SURVEY

In the paper "Intelligent eye: A mobile application for assisting blind people", published by **Milios Awad, Tarek Mahmoud**, et al., an Android-based mobile application is presented and showcased. It presents a user-friendly interface for blind people to navigate the environment using a single application. The application features colour detection, banknote recognition, object recognition and light detection. The paper also presents how the application can be more practical for the end-users and provide the desired features.

In the paper "A new Android application for blind and visually impaired people" by **Piotr Kardys, Damian Huderek**, et al., a novel approach to using voice commands is proposed to provide visually impaired people with some assistance while using the mobile application. It features some new capabilities, including sending, receiving text messages and using a phone book alongside monitoring their battery or positioning.

In the paper "BLIND GUARD: Designing Android Apps for Visually Impaired Persons", by **Mozibul Hoque, Monir Hossain**, et al., new features around the protection & security of visually impaired users are proposed. It showcases how easier it would be for visually impaired people to walk and speak with others while using the application. It also adds a new intelligent stick that can allow the visually impaired to use the application without any assistance and thus have new assistive features, like a location tracker and more. Alongside the new features, it can also help the visually impaired navigate by providing the best land route while travelling to a destination.

In the paper "BlindAid - Android-based Mobile Application Guide for Visually Challenged People" by **Gagani Senarathne, Hansi De Silva**, et al., the authors discuss three primary components: face recognition and obstacle identification and critical navigation. These components are proposed to drive a seamless user experience for the users and help them understand the environment better.

The paper "Image Captioning Methods and Metrics" by **Omkar Sargar and Shakti Kinger** was used to understand image captioning better and how Computer Vision (CV) and Natural Language Processing (NLP). Image Captioning is a method for extracting meaningful information from images, and it is made possible by identifying objects, actions, and their relationships and generating a description for the image. Captioning has been used across various fields: self-driving cars, assistive technology, medical imaging, etc. The paper determines the approach for image caption generation using the Convolutional Neural Network (CNN) and Generative Adversarial Network (GAN) model in a deep learning framework. We have adopted this methodology to develop an encoder-decoder architecture. Herein CNN is utilized for image vector generation, and LSTM is used to generate a logical sentence using the NLP modelling done previously. The paper also provides a detailed description of the architecture and the results obtained.

The paper "Real-Time Object Detection Android Application using Machine Learning Concepts for Visually Impaired People" by **Anil Thapa, Nilesh Wani**, et al. discussed Machine Learning and Google's TensorFlow Lite object detection API that provides a flexible and fast solution for accessing TensorFlow Lite interpreter and performing inference. We have utilized the proposed machine learning workflow to develop our workflow for building offline-first machine learning models.

CHAPTER 3

SCOPE AND REQUIREMENT ANALYSIS OF PRESENT INVESTIGATION

In this chapter, the proposed system has been compared with the existing system. The software and hardware requirements are provided.

3.1 EXISTING SYSTEM

The applications featured a clunky user interface that wasn't helpful to visually impaired people in the existing system. After extensive user research, we discovered that speech-controlled inferences are usually not so good for people with visual impairments and might lead to misunderstandings, thus putting the user in jeopardy. Additionally, the machine learning models proposed by the researchers were not so efficient for real-time inference, mainly while serving the machine learning models through a RESTful API. The image captioning system proposed by the researchers was not so efficient either and suffered network latency. It leads to delayed responses and thus brings regression problems.

3.2 PROPOSED SYSTEM

The project proposes the development of a new full-fledged mobile application to help visually impaired people with their daily activities. The application features a seamless, easy-to-use & intuitive user interface with offline-first capabilities. It allows the visually impaired people to navigate through the application quickly and perform tasks such as travelling, inferencing their environment, getting in touch for help, identifying text & currency notes and more. The mobile application has been put into real-life production and has been made available to a group of beta testers who have validated the efficacy of the application.

3.3 TECHNOLOGY USED

Visual Studio Code has been used as the software development environment to implement the project. Other technologies that have been used are Flutter, a cross-platform development framework for building our application, Flask for serving our machine-learning models, Tensorflow for our machine-learning models, and various third-party open-source libraries for integrating various features into our application.

3.4 REQUIREMENT ANALYSIS

Requirement analysis determines the requirements of a new system. This project analyses on product and resource requirement, which is required for this successful system. The product requirement includes input and output requirements it gives the wants in term of input to produce the required output. The resource requirements give in brief about the software and hardware that are needed to achieve the required functionality.

3.4.1 *Hardware Environment*

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It shows what the systems do and not how it should be implemented.

Hard Disk	:	1 TB and above
RAM	:	8GB and Above
Processor	:	macOS M1 Max
Active internet connection		

3.4.2 *Software Environment*

The software requirements are the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the team's and tracking the team's progress throughout the development activity.

- Platform : macOS M1 / Windows 10
- Front End : Flutter, Dart
- Back End : Flask
- Language : Python 3.9
- Software : Visual Studio Code, Jupyter Notebook
- Modules : Tensorflow, Keras, NumPy, Pandas, SciPy
- Environment : Android emulator
- Cloud Computing : Azure

CHAPTER 4

METHODOLOGY AND ALGORITHMS USED

In this chapter, architecture, workflow diagram with implementation and algorithms used has been discussed in detail.

4.1 INTRODUCTION TO MACHINE LEARNING

Machine learning is a field of computer science that uses statistical techniques to give computer systems the ability to "learn" (e.g., progressively improve performance on a specific task) with data, without being explicitly programmed.

The name machine learning was coined in 1959 by Arthur Samuel. Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data – such algorithms overcome following strictly static program instructions by making data-driven predictions or decisions, through building a model from sample inputs. Machine learning is employed in a range of computing tasks where designing and programming explicit algorithms with good performance is difficult or unfeasible; example applications include email filtering, detection of network intruders, and computer vision.

Machine learning is closely related to (and often overlaps with) computational statistics, which also focuses on prediction-making through the use of computers. It has strong ties to mathematical optimization, which delivers methods, theory and application domains to the field. Machine learning is sometimes conflated with data

mining, where the latter sub-field focuses more on exploratory data analysis and is known as unsupervised learning.

Within the field of data analytics, machine learning is a method used to devise complex models and algorithms that lend themselves to prediction; in commercial use, this is known as predictive analytics. These analytical models allow researchers, data scientists, engineers, and analysts to "produce reliable, repeatable decisions and results" and uncover "hidden insights" through learning from historical relationships and trends in the data.

4.2 Introduction to Deep Learning & Transfer Learning

The area of Deep Learning traces back to the 1940's where artificial intelligence research was about to come to fruition. In 1943, neuroscientists Warren McCulloch and Walter Pitts proposed an artificial neuron known as the McCulloch-Pitts (MCP) neuron. This neuron formed the basis of the first mathematical model of an artificial neuron.

During the 1980s and 1990's adding multiple layers to neural networks were showing promising results and led to breakthroughs in deep learning. The Neocognitron was one of those promising models proposed by Kunihiro Fukushima in 1980. The Convolutional Neural Network (CNN) was introduced by LeCun, Boser, et al. where it was applied on handwritten digits with a 1% error rate.

A year before the CNN, another neural network, the Long Short-Term Memory (LSTM) Neural Network was invented to solve a specific problem. LSTM's is an evolved version of the Recurrent Neural Network; a network which has cycles, giving it the ability to handle sequential data one element at a time. LSTM's are designed to address this problem; by introducing a "memory cell" and gated units, enabling the network to remember information when it needs to selectively. The benefit is that

previous sequences are remembered for an extended period without degradation, as opposed to the RNN.

Convolutional Neural Networks

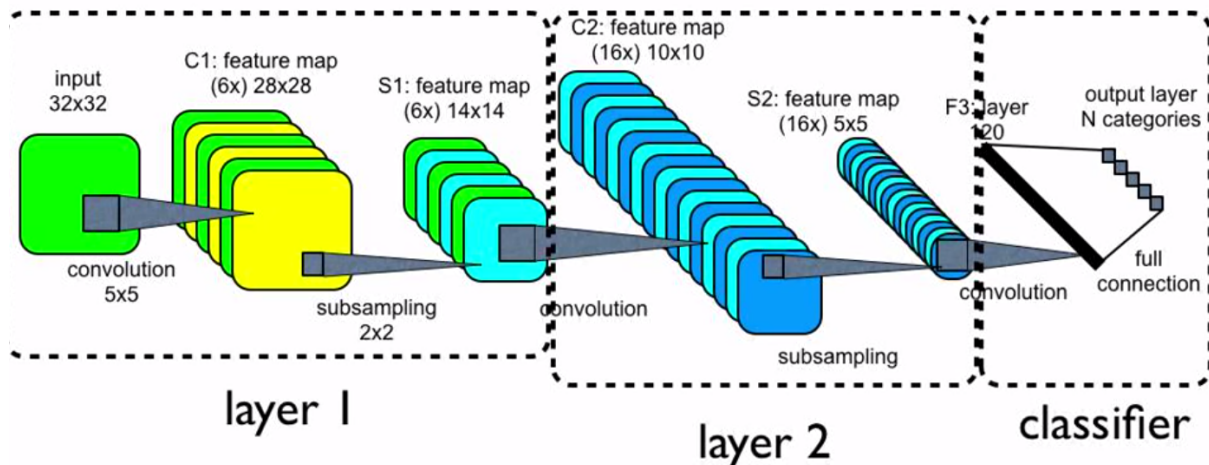


Fig 4.1: Explaining Convolutional Neural Networks

Transfer Learning was heralded with the rise of Pre-Trained Models that came up since 2012, after AlexNet revolutionized the Machine Learning World. AlexNet was adjudged the winner of ImageNet Large Scale Visual Recognition Challenge and this started a race among Machine Learning Researchers to build Models that can outperform AlexNet.

After AlexNet, Inception and Resonant were born, which further heralded a believe that by taking advantages of pre-trained Neural Networks which are fine-tuned and modelled accurately, can be used by Machine Learning Researchers for their own purpose. This led to the rise of an idea known as Transfer learning. The idea behind

transfer learning is that a neural network that has been trained on a large dataset can apply its knowledge to a dataset it has never seen before.

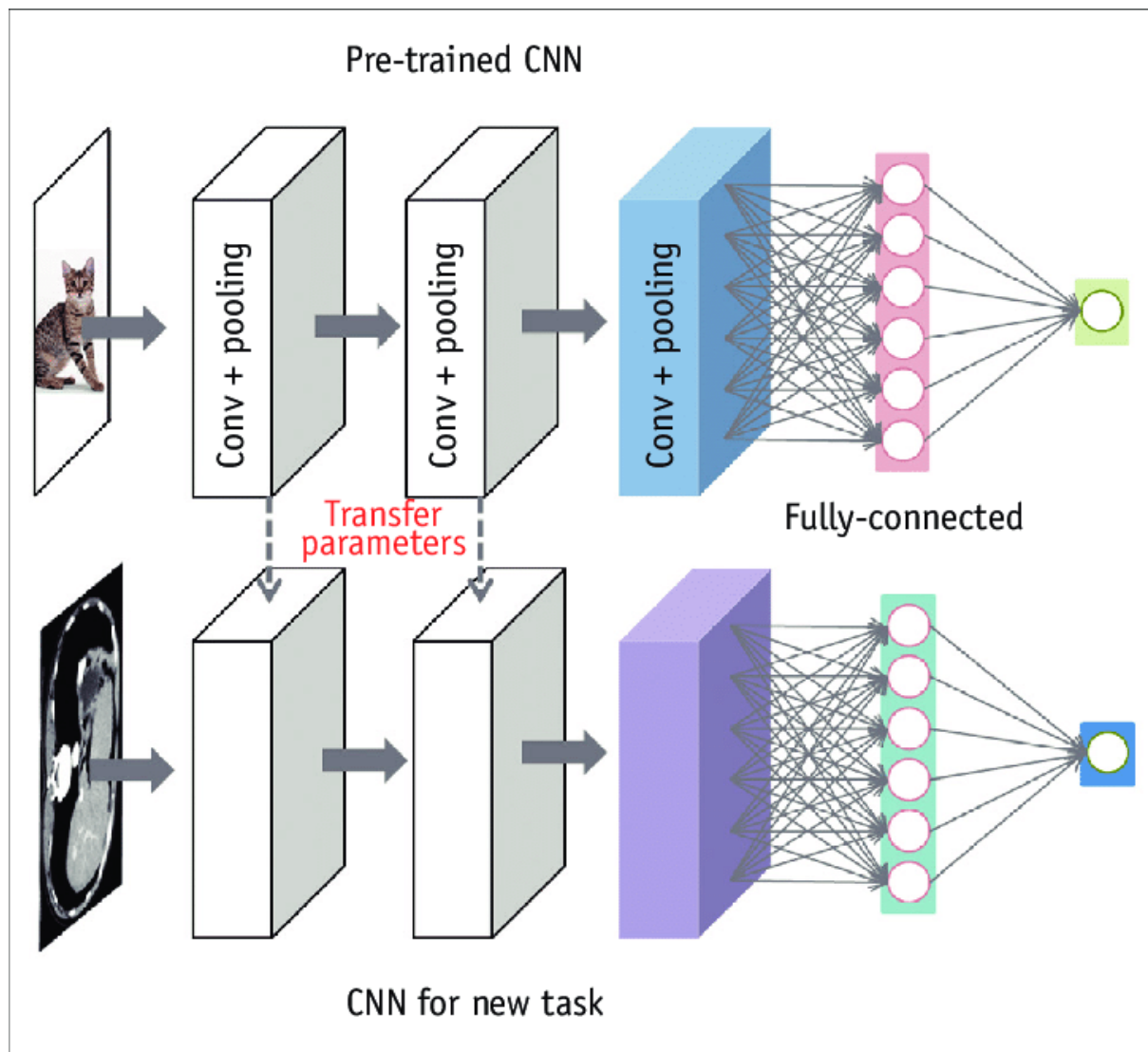


Fig 2: Explaining Transfer Learning

In order to perform Transfer Learning, we just need to change the last layer of the Pre-Trained Model. For example, the ImageNet Dataset has 1000 Output Classes and in comparison, the Fashion MNIST has got only 10 Classes. Therefore, we only

need to change the last layer of our existing model so that it has the same number of classes that we have in our dataset. We also need to render the pre-trained model as “non-trainable” hence effectively freezing them so that we don’t accidentally train them during the process.

By glancing at the history of deep learning, it is apparent that there is promise in its potential to solve numerous problems. With inspiration from biology and in combination with the computational power of GPUs, neural networks today are being actively researched.

4.3 Technologies used

Flutter

Flutter is an open-source UI software development kit created by Google. It is used to develop cross platform applications for Android, iOS, Linux, Mac, Windows, Google Fuchsia, Web platform, and the web from a single codebase. The Flutter SDK contains all the elements we need to perform Flutter development:

- **Dart Platform:** A programming language designed for client development such as for the web and mobile apps.
- **Flutter engine:** Runtime on which Flutter applications run. It provides graphic rendering support, as well as providing an interface (through the Flutter core libraries) to the Android or iOS SDK.
- **Foundation Library:** The Foundation library, written in Dart, provides basic classes and functions which are used to construct applications using Flutter, such as APIs to communicate with the engine.

Python

Python is a general-purpose Programming Language that is utilized for a variety of tasks which include: Web Development, API Development, Data Modelling, Statistical Computing and General-Purpose Development. Python was developed by Guido van Rossum and is named after Monty Python's Flying Circus. Python has got many features which makes it comprehensible, easy to understand and to implement them as well. These features include:

- **Dead Simple Syntax:** Python features one of the easiest syntax for any Programming Language. It is a High-Level Language which allows many beginner programmers to get started with Python easily. The syntax is expressive and is easy to understand and hence learn as well.
- **Cross-Platform and Open-Source:** The biggest advantage of Python is that it is Open-Source and is available on Cross-Platform. The same Python code can be executed on different hardware and Operating System without any hassle. Python is also Open-Source and hence it can be utilized without any operational costs.
- **Large Number of Libraries:** Python has got a large number of Third-Party Libraries and Modules which makes it easy to work with. Even in our current scope of Project, we will be making use of many Libraries and Modules for our Purpose.

In our current Project, we made use of Python for:

- **Data Modelling:** With the aid of Third-Party Libraries and Modules, we will be using Python for creating our Model Architecture.
- **Model Deployment:** With the aid of Flask, a Micro Web-Service Framework we will be deploying the Model in a RESTful API Architecture.

- **Data Pre-Processing:** With the aid of Python and third-party modules, we will be pre-processing the Dataset available to us to make it fit for the Data Modelling and Deployment.

Tensorflow

Tensorflow is an open source machine learning framework owned and maintained by Google. It is used for implementing machine learning and deep learning applications. It combines the computational algebra of optimization techniques for easy calculation of many mathematical expressions. Tensorflow has got many features which makes it comprehensible, easy to understand and to implement Deep Learning Models as well. These features include:

- It includes a unique feature that defines, optimizes and calculates mathematical expressions easily with the help of multi-dimensional arrays called tensors.
- It includes programming support of deep neural networks and machine learning techniques.
- It enables high scalable feature of computation with various data sets.
- Tensorflow uses GPU computing, automating management. It also includes a unique feature of optimization of same memory and the data used.

Tensorflow is implemented using C++ and has got high-level APIs which are accessible through Keras, which we will be using for our purpose. We will be using Tensorflow and Keras for the following purposes:

- **Data Pre-Processing:** With the aid of Keras Pre-Processing, we will be pre-processing the Dataset before we move ahead with the Data Deployment part of the Application.

- **Data Modelling:** With the aid of Keras, we will be building a customized deep learning architecture written from the scratch which can be then used for the image captioning & object detection.
- **Transfer Learning:** With the aid of Keras Applications, we will be making use of pre-trained weights from large models for the purpose of our Classification Problem.

Flask

Flask is a Micro Web-Framework that is used for developing Backend Interface for your Application. It is a lightweight framework which has a small footprint and can be used to set up things pretty quickly, with a simple yet extensible core.

Flask was created by Armin Ronacher of Pocoo, an international group of Python enthusiasts formed in 2004. It is designed to help getting started quick and easy, with the ability to scale up to complex applications. It began as a simple wrapper around Werkzeug and Jinja and has become one of the most popular Python web application frameworks.

Flask is currently being used to develop RESTful APIs that has got multiple endpoints and is seen as a fast and reliable way to deploy Machine Learning Models. Some of its features include:

- **Easy to Understand:** Flask is very easy to understand and to test. Using Flask, we can setup routes for our Backend Service as easy as writing Python functions.
- **Third-Party Modules:** Flask supports many Third-Party Modules which enhances the modularity of the Code and allows the Developers to scale it as they like.

- **Integrated Development Server and Debugging:** The biggest advantage of Flask includes its integrated Development Server which allows us to set it up on a Local Machine and debug our application.

Flask was utilized by us for developing the Application Programming Interface (API) for the application, by following the RESTful standards of development. A single endpoint was exposed, which loads the H5 file for generating inferences.

NumPy

NumPy is a library for Python, adding support for large, multidimensional arrays and matrices, along with a large collection of highlevel mathematical functions to operate on these arrays. NumPy is an open-source software and has many contributors.

NumPy targets the CPython reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents. NumPy addresses the slowness problem partly by providing multidimensional arrays and functions and operators that operate efficiently on arrays, requiring rewriting some code, mostly inner loops using NumPy.

Using NumPy in Python gives functionality comparable to MATLAB since they are both interpreted, and they both allow the user to write fast programs as long as most operations work on arrays or matrices instead of scalars. In comparison, MATLAB boasts a large number of additional toolboxes, notably Simulink, whereas NumPy is intrinsically integrated with Python, a more modern and complete programming language. Moreover, complementary Python packages are available; SciPy is a library that adds more MATLAB-like functionality and Matplotlib is a plotting package that provides MATLAB-like plotting functionality.

Pandas

Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with structured (tabular, multidimensional, potentially heterogeneous) and time series data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open-source data analysis / manipulation tool available in any language. It is already well on its way toward this goal.

The two primary data structures of pandas, Series (1- dimensional) and DataFrame (2-dimensional), handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering. For R users, DataFrame provides everything that R's dataframe provides and much more. pandas are built on top of NumPy and is intended to integrate well within a scientific computing environment with many other 3rd party libraries.

Others

- **Anaconda:** Anaconda is a popular data science platform where you can create data science projects and machine learning. Libraries such as NumPy, Pandas, Matplotlib, Tensorflow and etc come with Anaconda and IDE's such as Jupyter Notebook, Spyder and etc.
- **Keras:** Keras is also a Deep Learning Framework that abstracts much of the code in the other Frameworks like Tensorflow and Theano. Compared to the other frameworks Keras is more minimalist.
- **Jupyter Notebook IDE:** The Anaconda distribution comes with a variety of software that includes Jupyter Notebooks for scientific computing. Jupyter Notebooks is an open source software IDE that allows developers to create and share documents that contain live code and more.
- **Azure:** Azure is Microsoft's public cloud computing platform. It provides a range of cloud services, including compute, analytics, storage and networking.

We are using Azure to deploy our Flask APIs and corresponding machine learning models on the cloud.

4.4 Image Captioning model

Training models to be able to automatically generate descriptive captions for images is currently a very hot topic in Computer Vision and Machine Learning. This is a combination of image scene understanding, feature extraction, and translation of visual representations into natural languages.

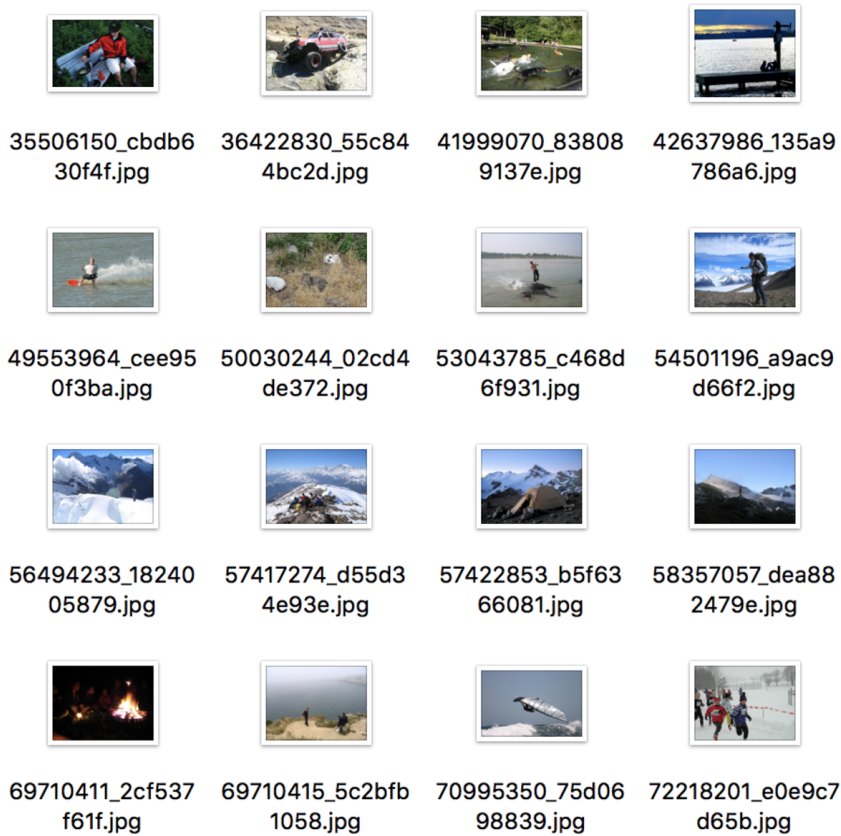


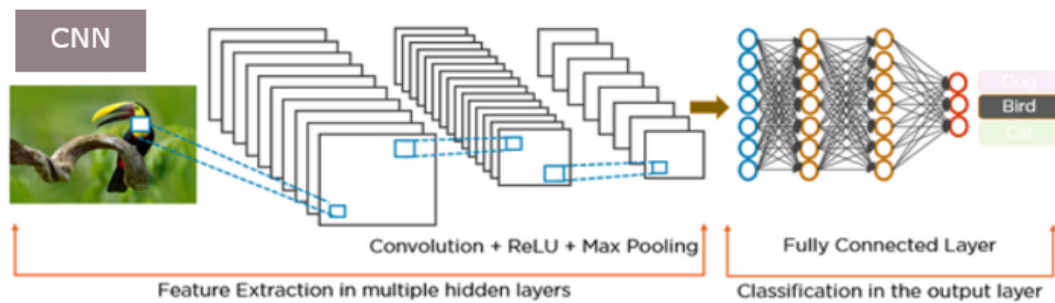
Fig 3: Flickr8k dataset

We have written data preprocessing scripts to process raw input data (both images and captions) into proper format; A pre-trained Convolutional Neural Network

architecture as an encoder to extract and encode image features into a higher dimensional vector space; An LSTM-based Recurrent Neural Network as a decoder to convert encoded features to natural language descriptions.

We have identified the three most commonly used image caption training datasets in Computer Vision research domain - COCO dataset, Flickr8k and Flickr30k. These datasets contain 123,000, 31,000 and 8,000 caption annotated images respectively and each image is labeled with 5 different descriptions. Currently, Flickr8k dataset which contains the least number of images is used as our primary data source over the other two due to our limited storage and computational power.

Convolutional Neural Network



Recurrent Neural Network

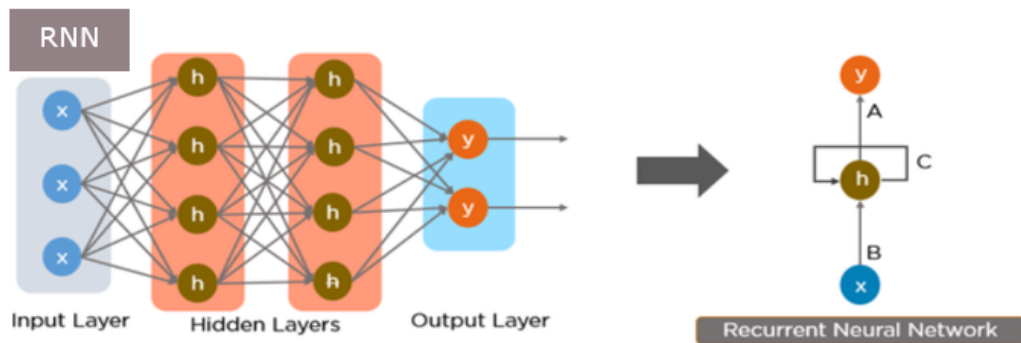


Fig 4: Recurrent Neural Networks and Convolutional Neural Networks

Input data are composed of images and captions, and hence we need to pre-process both the images into proper format for CNN network and the text captions into proper format for RNN network. Since our image caption generator pipeline is leveraging pre-trained state-of-the-art CNN network, we need to transform images into correct format.

The encoder needs to extract image features of various sizes and encodes them into vector space which can be fed to RNN in a later stage. VGG-16 and ResNet is commonly recommended as image encoders. We chose to modify the pre-trained VGG-16 model with CNN used to encode features instead of classify images. As a result, we removed the fully connected layers and the max pool layers at the end of the network. Under this new construction, the input image matrix has dimension $N \times 3 \times 256 \times 256$, and the output has dimension $N \times 14 \times 14 \times 512$. Furthermore, in order to support input images with various sizes, we added an adaptive 2d layer to our CNN architecture.

Following the CNN, we built the soft trainable Attention mechanism introduced in Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. Attention mechanism tells the network which part of the image should be focused on for generating the next word in the description. We calculated the attention area through adding the encoder output and historical state, which is updated in each iteration.

In implementing the attention layer, we performed linear transformation on both the encoder output and historical state output. The linear activated outputs are summed up and the activation function is ReLU. The weight(alpha) and the attention area (applying alpha on encoder output) are returned. The decoder needs to generate image captions word by word using a Recurrent Neural Network - LSTMs which is able to sequentially generate words. The input for the decoder is the encoded image feature vectors from CNN and the encoded image captions produced in data preprocessing stage.

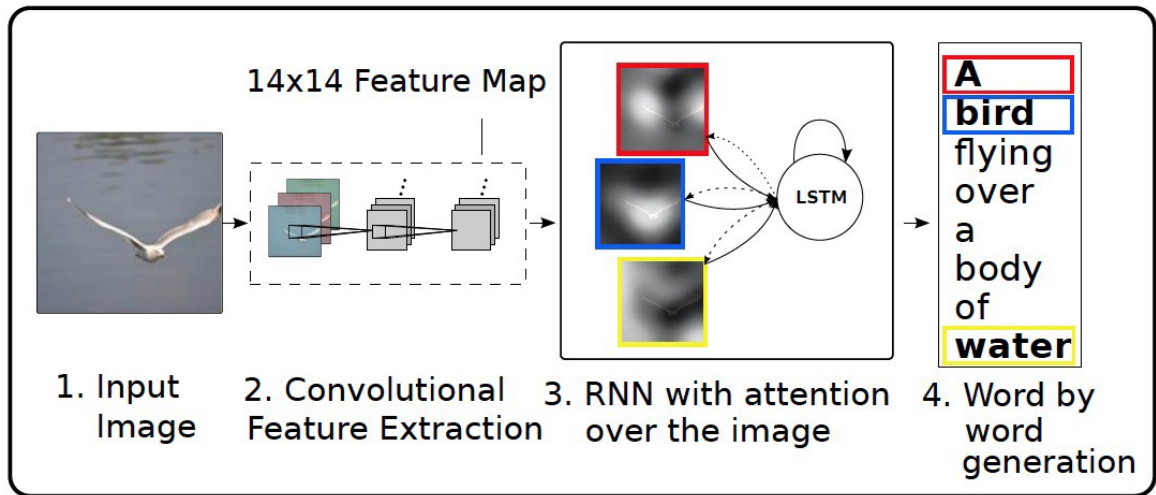


Fig 5: Image Caption generation using CNN and RNN

In each iteration of LSTM network, we first put the historical state of LSTM and the encoded images into the Attention module to get the attention-masked images which have a specific highlighted area. Then we concatenate the embedded captions of all previous words and the attentioned images and feed them to the LSTM to get the next state of LSTM. Then, fully connected layers can predict the probabilities of current word embedding based on the current state and append it to the word embedding prediction matrix.

Herein the data is encoded to numbers via a dictionary of words. After the data is consumed by the pipeline, the output will also be in the encoded format which needs to be reversed back into English words in order to make sense to human. Some of the results that we received from our model has been showcased below.

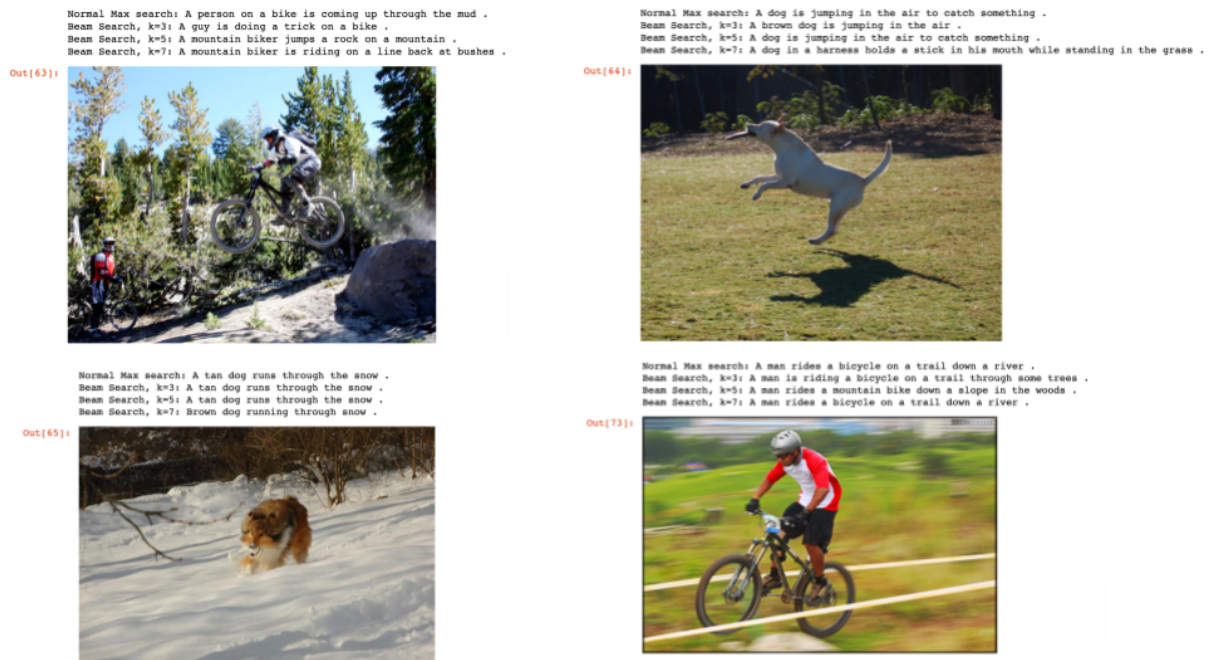


Fig 6: Image Caption generation in action

4.5 Object detection model

Object detection is very common in computer vision and image processing. It can detect a certain class of objects (such as people, vehicles) the image and locate the object. Several approaches (R-CNN, Fast R-CNN, Faster R-CNN, SSD) have been applied to many fields such as outdoor robotics, autonomous driving and computer vision. Before SSD (single-shot detection), there are three kinds of available approaches (R-CNN, Fast R-CNN, Faster R-CNN). All of them are slow, although the fastest one Faster R-CNN can achieve 7FPS (detect seven images per second). So, it can hardly achieve real-time object detection. SSD is faster than the state-of-the-art single shot detector. SSD is the first object detector based on deep learning that does not resample the image pixels.

Deep learning is effective in tasks such as Image Classification, Object Detection, and Natural Language Processing. Generally, the complexity of the model often limits its deployment in a variety of real-world application scenarios and resource-constrained edge devices. In order to apply SSD on the mobile device, to eliminate the storage size and achieve faster inference time are significant.

In TensorFlow, quantization is work with fake quantization node. Quantization - aware training. The training of the quantitative perception model ensures that the forward transfer matches the accuracy of the training and reasoning which shows in figure one. There are two aspects to this: Operator fusion in reasoning is precisely modeled during training the quantitative effects of reasoning are modeled during training.

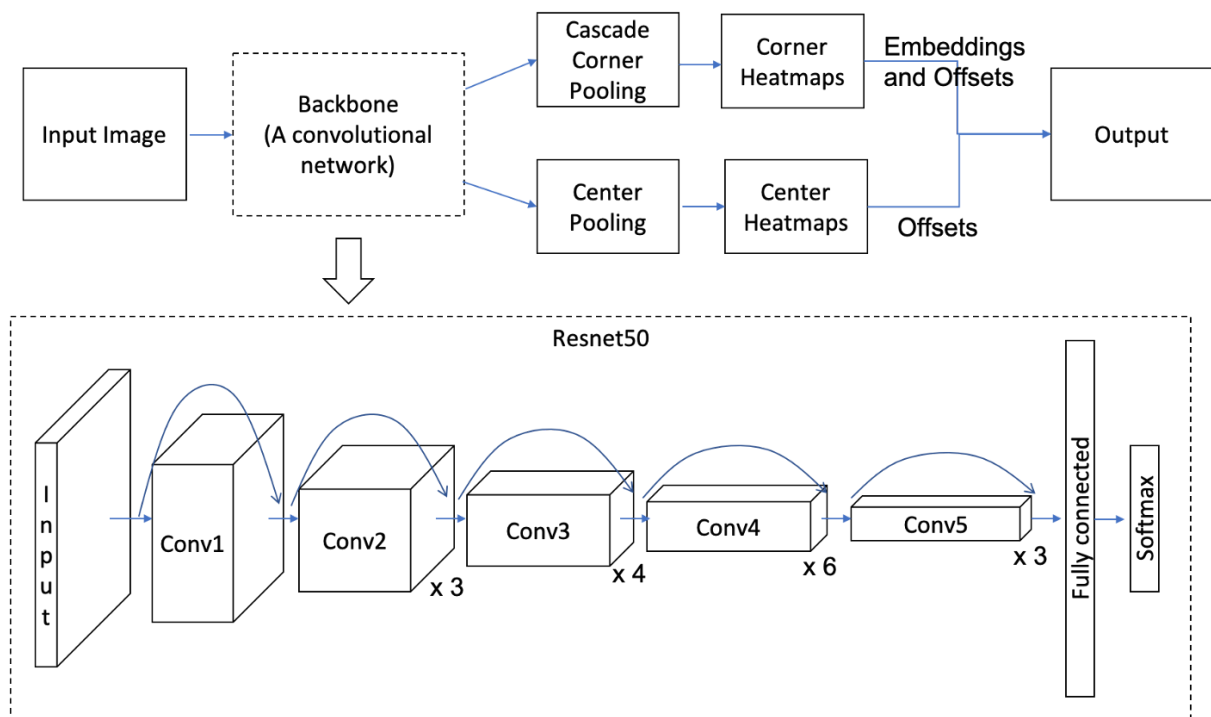


Fig 7: Object detection model in Tensorflow

In order to carry out effective reasoning, TensorFlow combines the batch normalization with the previous convolution and full connection layer by folding the

batch norm layer and then quantifies it. The quantization error is modeled by using a pseudo quantization node, and the quantization effect of forwarding and backward channels is simulated.

The forward transfer model is quantized, and the backward transfer model is quantized as a straightthrough estimation. The quantization of weights and activities is simulated in both forward and backward transfers. In the process of backpropagation, parameters need to be updated with high precision to ensure accuracy when accumulating minor adjustments of parameters. In addition, the minimum and maximum activation values are determined during training. This allows models that have been quantitatively trained in the loop to be easily converted into fixedpoint inference models, eliminating the need for a separate calibration step.

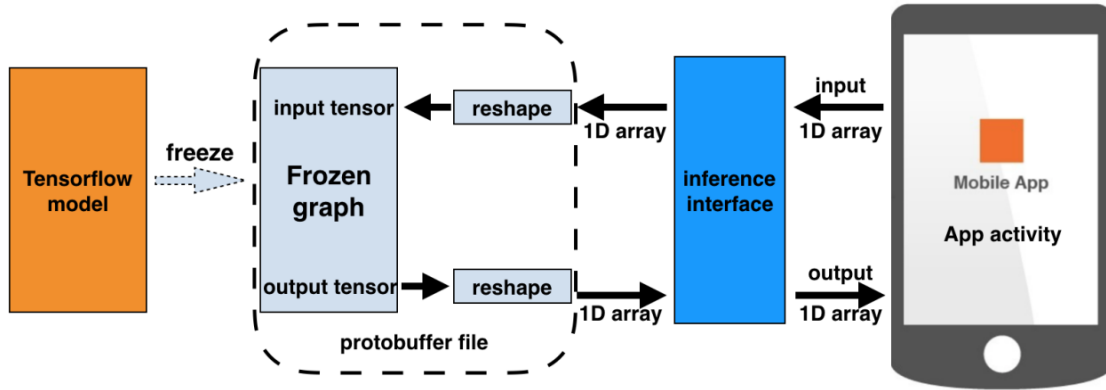


Fig 8: Integration of Tensorflow Lite

We comprehensively evaluate our method on the MSCOCO dataset. This dataset consists of about 35 categories and 7.7 instance targets per image, less than 20% of images contain only one category, and only 10% of images contain only one instance target. The images in MSCOCO contain natural pictures and common target pictures in life. The background is relatively complex, the number of targets is

large, and the size of targets is smaller. Some of the results that we received from our model has been showcased below.

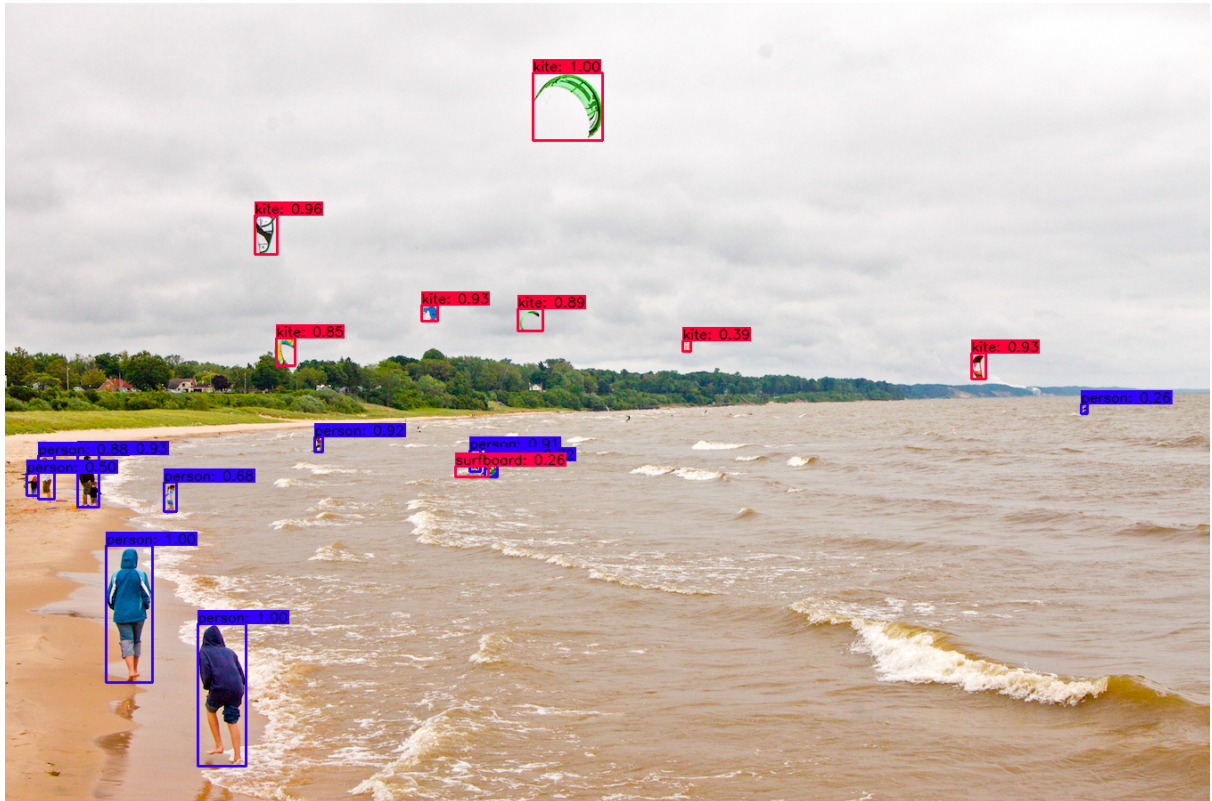


Fig 9: Tensorflow Object Detection in action

4.6 SYSTEM ARCHITECTURE

The main features of the application are:

- **Live Image Labelling:** Live Image Labelling is one of the features that capitalize on Tensorflow. To make this possible we are capitalizing on the capability of Transfer Learning to develop the Models real quick and to get an appreciable Performance Metric even in the lacking support for high-ended

devices.

- Currency Detection: Our application allows users to detect currency notes as well. We have used a custom trained tflite model trained on Indian currency.
- Text Extraction: Users might sometimes need to read text present in a photo they clicked, or have saved in their gallery. Being visually impaired, it might not be easy for them to properly read the text. So we have provided a feature for users to click a photo, select existing photos, and extract all the text from them. This text is then spoken out to the user using our text-to-speech functionality.
- Image Captioning: Image Captioning is used to provide a live feed about the surroundings and the activities happening around visually impaired people. It has been developed using Tensorflow and Flask and extended as an API that can be plugged into our application for ease of use.
- SOS Feature: One of the most important features of the application is the SOS feature. In order to use the feature, a user just needs to enter 3 phone numbers on their phone. To access the feature immediately without much hassle, a user just needs to vigorously shake the phone for 3-4 seconds and an SOS message with the exact location details of the user would be sent to the mobile numbers the user has entered.

Here is the architecture diagram for all of the aforementioned features:

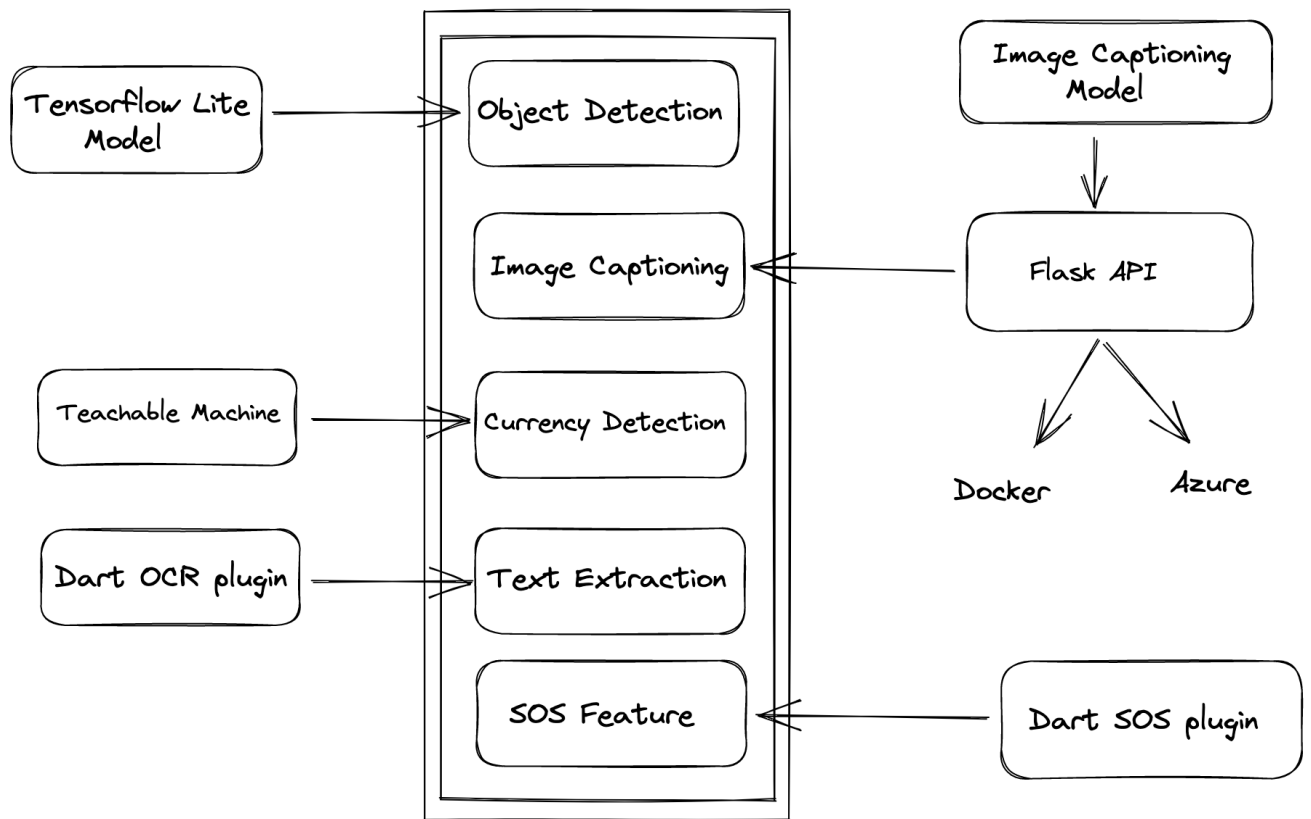


Fig 10: System architecture diagram

CHAPTER 5

RESULTS AND DISCUSSION

This section present some results showing the efficiency of the Helping Hands mobile application in detecting the detecting objects, recognizing banknotes, live image captioning images and more:

Live image labelling:

We are making use of the SSD Mobilenet Model from Keras and set the Trainable Layers to False so that we don't train them from our side. After successfully modelling the Data, we can convert them to a Tflite Model that can be served using a Tflite Plugin.

The Tflite Plugin will be induced as a Dev Dependency that will be used to integrate our neural network with the app. The Tflite Plugin will be used with a loadModel() function which would load the Image and then use the runModelOnFrame() function to take the Camera Stream Input and generate the inferences henceforth.

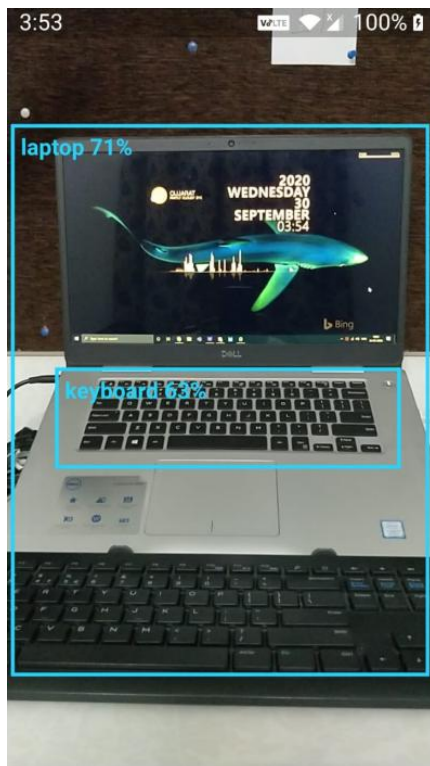


Fig 11: Detecting objects on mobile application

Currency Detection

Open the currency detection screen and tap on it. Click a photo of the note. Our model can classify low quality or blurred images, or folded notes as well with high accuracy. The model will classify the image, show a dialogue along with the prediction. For better accessibility, our text-to-speech functionality will also speak out the prediction made so the user does not need to actually see the screen for the prediction made.

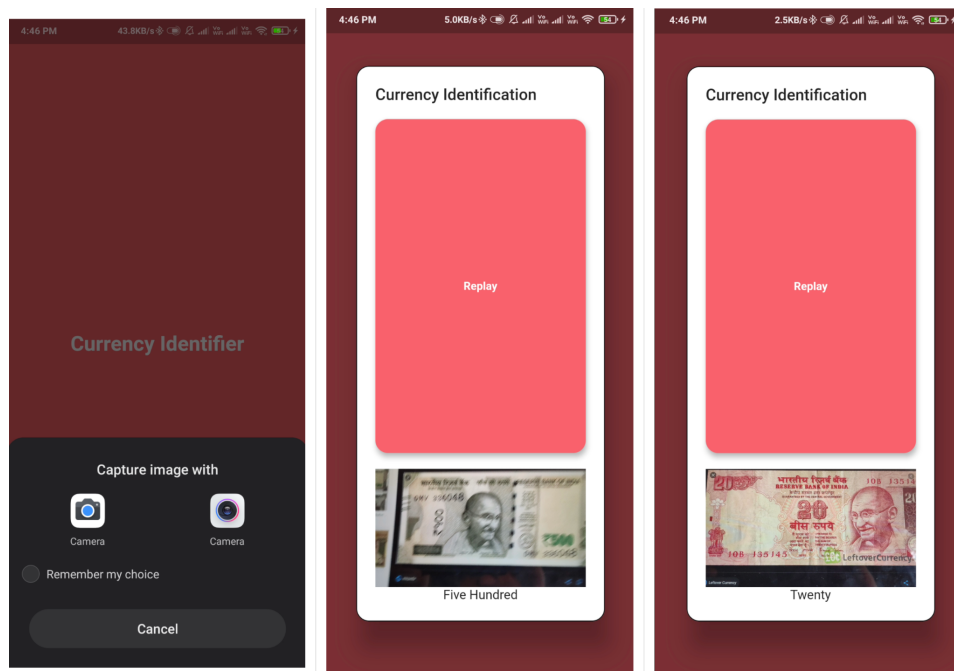


Fig 11: Detecting currency notes on the application

Text Extraction

Open the Text Extraction screen and tap on it. Click/Select a photo. The photo, as well as the text extracted using OCR functionality, is displayed in a dialogue, with stop, replay, and pause options and is spoken out to the user as well.

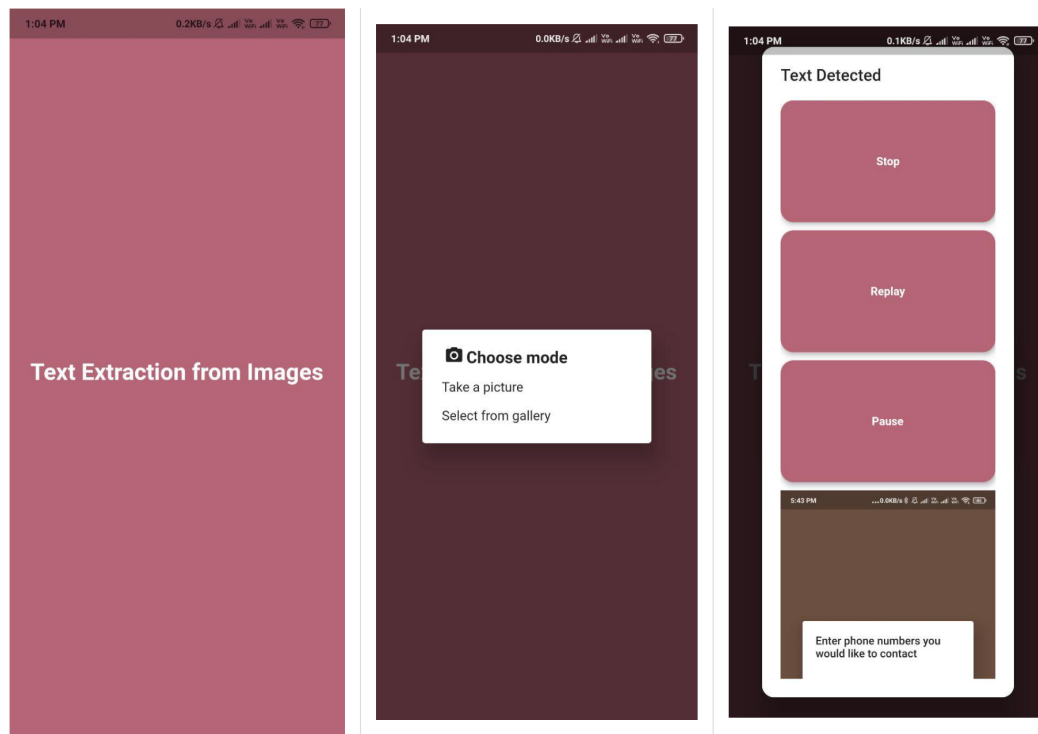


Fig 11: Detecting text from images on the application

User acceptability

Furthermore, the application was tested by 10 people with visible impairments. Each person had to answer a set of questions. For each question, the person had to provide an integer number between 1 and 5, with 1 meaning "poor performance" and 5 meaning "excellent performance". In addition, the questionnaire allowed each person to provide comments on the application. It can be seen from the table below that the results are good in general and the application is well accepted.

	Average	Median	Min	Max
Familiarity of the interface	4.4	4.5	3	5
How easy is the application as a whole	4.1	4	3	5
How good is the performance	3.7	4	3	4
How is its responsiveness	4	4	3	5
Clarity of the prompt messages	3.9	4	3	5

CHAPTER 6

CONCLUSION AND FUTURE WORK

6.1 CONCLUSION

In this report, we introduced the Helping Hands mobile application. The application targets visually impaired people and provide them with a set of useful features such as image captioning, object recognition, text extraction and banknote recognition. It has a user-friendly interface customized for blind people, where the detection results are read out loud so that the user can clearly hear them. Results showed that the application correctly detected objects, environment, currency notes and image texts.

6.2 FUTURE WORK

Future enhancements include, but are not limited to, adding more features such as Scanning and reading barcodes, personal card reader, GPS reader, pedestrian guide, and traffic light detector. The application would also be subjected to more rigorous testing with visually impaired people and a peer-reviewed research work would be published to determine the efficacy of the machine learning models and the application flow.

REFERENCES

- [1] M. Awad, J. E. Haddad, E. Khneisser, T. Mahmoud, E. Yaacoub and M. Malli, "Intelligent eye: A mobile application for assisting blind people," *2018 IEEE Middle East and North Africa Communications Conference (MENACOMM)*, 2018, pp. 1-6, doi: 10.1109/MENACOMM.2018.8371005.
- [2] P. Kardyś, A. Dąbrowski, M. Iwanowski and D. Huderek, "A new Android application for blind and visually impaired people," *2016 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA)*, 2016, pp. 152-155, doi: 10.1109/SPA.2016.7763604.
- [3] G. Senarathne, D. Punchihewa, D. I. Liyanage, G. Wimalaratne and H. De Silva, "BlindAid - Android-based Mobile Application Guide for Visually Challenged People," *2021 IEEE 12th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, 2021, pp. 0039-0045, doi: 10.1109/IEMCON53756.2021.9623189.
- [4] M. Hoque, M. Hossain, M. Y. Ali, N. Nessa Moon, A. Mariam and M. R. Islam, "BLIND GUARD: Designing Android Apps for Visually Impaired Persons," *2021 International Conference on Artificial Intelligence and Computer Science*

- Technology (ICAICST), 2021, pp. 217-222, doi: 10.1109/ICAICST53116.2021.9497799.
- [5] S. A. Jakhete, P. Bagmar, A. Dorle, A. Rajurkar and P. Pimplikar, "Object Recognition App for Visually Impaired," 2019 IEEE Pune Section International Conference (PuneCon), 2019, pp. 1-4, doi: 10.1109/PuneCon46936.2019.9105670.
- [6] Xu K, CA U, Ba JL, CA U, Kiros R, EDU T, Cho K, Courville A, Salakhutdinov R, Zemel RS, Bengio Y. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention.
- [7] N. E. Albayrak, "Object Recognition using TensorFlow," 2020 IEEE Integrated STEM Education Conference (ISEC), 2020, pp. 1-1, doi: 10.1109/ISEC49744.2020.9397835.
- [8] B. N. K. Sai and T. Sasikala, "Object Detection and Count of Objects in Image using Tensor Flow Object Detection API," 2019 International Conference on Smart Systems and Inventive Technology (ICSSIT), 2019, pp. 542-546, doi: 10.1109/ICSSIT46314.2019.8987942.
- [9] B. N. K. Sai and T. Sasikala, "Object Detection and Count of Objects in Image using Tensor Flow Object Detection API," 2019 International Conference on Smart Systems and Inventive Technology (ICSSIT), 2019, pp. 542-546, doi: 10.1109/ICSSIT46314.2019.8987942.
- [10] S. Boukhary and E. Colmenares, "A Clean Approach to Flutter Development through the Flutter Clean Architecture Package," 2019 International Conference on Computational Science and Computational Intelligence (CSCI), 2019, pp. 1115-1120, doi: 10.1109/CSCI49370.2019.00211.

APPENDIX

A. Source Code

i) main.dart

```
import 'dart:async';

import 'package:flutter/material.dart';

import 'package:camera/camera.dart';

import 'package:helping_hands/splashscreen.dart';
```

```
List<CameraDescription> cameras;
```

```
Future<Null> main() async {

  WidgetsFlutterBinding.ensureInitialized();

  try {

    cameras = await availableCameras();

  } on CameraException catch (e) {

    print('Error: $e.code\nError Message: $e.message');

  }
```

```
runApp(new MyApp());  
}
```

```
class MyApp extends StatelessWidget {  
  
  @override  
  
  Widget build(BuildContext context) {  
  
    return MaterialApp(  
  
      title: 'Helping Hands',  
  
      debugShowCheckedModeBanner: false,  
  
      home: MySplash(),  
  
    );  
  
  }  
}
```

ii) home.dart

```
import 'dart:async';  
  
import 'dart:io';  
  
  
  
import 'package:flutter/material.dart';  
  
import 'package:camera/camera.dart';  
  
import 'package:flutter_icons/flutter_icons.dart';
```

```
import 'package:flutter_tts/flutter_tts.dart';

import 'package:geolocator/geolocator.dart';

import 'package:helping_hands/currency_detection/currency.dart';

import 'package:helping_hands/image_captioning/image_captioning.dart';

import 'package:helping_hands/ocr/dialog_ocr.dart';

import 'package:helping_hands/sos/sos_dialog.dart';

import 'package:hexcolor/hexcolor.dart';

import 'package:image_picker/image_picker.dart';

import 'package:shake/shake.dart';

import 'package:shared_preferences/shared_preferences.dart';

import 'package:simple_ocr_plugin/simple_ocr_plugin.dart';

import 'package:tflite/tflite.dart';

import 'package:telephony/telephony.dart';

import 'dart:math' as math;

import 'package:vibration/vibration.dart';


import 'live_labelling/bndbox.dart';

import 'live_labelling/camera.dart';


class HomePage extends StatefulWidget {
```

```
final List<CameraDescription> cameras;
```

```
HomePage(this.cameras);
```

```
@override
```

```
_HomePageState createState() => new _HomePageState();
```

```
}
```

```
class _HomePageState extends State<HomePage> {
```

```
  List<dynamic> _recognitions;
```

```
  int _imageHeight = 0;
```

```
  int _imageWidth = 0;
```

```
  String _model = "SSD MobileNet";
```

```
  File _capImage;
```

```
  File _currImage;
```

```
  final FlutterTts flutterTts = FlutterTts();
```

```
  final Telephony telephony = Telephony.instance;
```

```
  PageController _controller = PageController(
```

```
    initialPage: 0,
```

```
);
```

```
Future getCurrImage() async {
```

```
    final currImage = await ImagePicker.pickImage(source: ImageSource.camera);
```

```
    setState(() {
```

```
        _currImage = currImage;
```

```
    });
```

```
    if (currImage != null) {
```

```
        CurrPage.currencyDetect(context, currImage);
```

```
    }
```

```
}
```

```
Future getCapImage() async {
```

```
    final capImage = await ImagePicker.pickImage(source: ImageSource.camera);
```

```
    setState(() {
```

```
        _capImage = capImage;
```

```
    });
```

```
    if (capImage != null) {
```

```
        imgCap.uploadImg(context, _capImage);
```

```
    }
```

```
}
```

```
var sosCount = 0;
```

```
var initTime;
```

```
@override
```

```
Future<void> initState() {
```

```
    super.initState();
```

```
    smsPermission();
```

```
    loadModel();
```

```
    ShakeDetector detector = ShakeDetector.waitForStart(onPhoneShake: () {
```

```
        if (sosCount == 0) {
```

```
            initTime = DateTime.now();
```

```
            ++sosCount;
```

```
        } else {
```

```
            if (DateTime.now().difference(initTime).inSeconds < 4) {
```

```
                ++sosCount;
```

```
                if (sosCount == 6) {
```

```
                    sendSms();
```

```
                    sosCount = 0;
```

```

    }

    print(sosCount);

} else {

    sosCount = 0;

    print(sosCount);

}

}

});

```

```

    detector.startListening();

}

```

```

void sendSms() async {

    SharedPreferences prefs = await SharedPreferences.getInstance();

    String n1 = prefs.getString('n1');

    String n2 = prefs.getString('n2');

    String n3 = prefs.getString('n3');

    String name = prefs.getString('name');

    Position position =

        await getCurrentPosition(desiredAccuracy: LocationAccuracy.best);

```

```

if (position == null) {

    position = await getLastKnownPosition();

}

String lat = (position.latitude).toString();

String long = (position.longitude).toString();

String alt = (position.altitude).toString();

String speed = (position.speed).toString();

String timestamp = (position.timestamp).toISOString();

print(n2);

telephony.sendSms(

    to: n1,

    message:

        "$name needs you help, last seen at: Latitude: $lat, Longitude: $long,
Altitude: $alt, Speed: $speed, Time: $timestamp");

telephony.sendSms(

    to: n2,

    message:

        "$name needs you help, last seen at: Latitude: $lat, Longitude: $long,
Altitude: $alt, Speed: $speed, Time: $timestamp");

telephony.sendSms(

```

```

    to: n3,

    message:

        "$name needs you help, last seen at: Latitude: $lat, Longitude: $long,
        Altitude: $alt, Speed: $speed, Time: $timestamp");

}

```

```

void smsPermission() async {

    //bool permissionsGranted = await telephony.requestPhoneAndSmsPermissions;

}

```

```

loadModel() async {

    String res = await Tflite.loadModel(

        model: "assets/ssd_mobilenet.tflite",

        labels: "assets/ssd_mobilenet.txt");

    print("MODEL" + res);

}

```

```

setRecognitions(recognitions, imageHeight, imageWidth) {

    setState(() {

        _recognitions = recognitions;
    });
}

```

```

    _imageHeight = imageHeight;

    _imageWidth = imageWidth;

  });
}

```

@override

```

Widget build(BuildContext context) {

  Size screen = MediaQuery.of(context).size;

  sosDialog sd = new sosDialog();

  ocrDialog od = new ocrDialog();

  return Scaffold(

    body: PageView(

      controller: _controller,

      onPageChanged: _speakPage,

      children: <Widget>[

        Container(

          child: Stack(

            children: [

              Camera(

                widget.cameras,

```

```

        _model,

        setRecognitions,

    ),

    BndBox(

        _recognitions == null ? [] : _recognitions,

        math.max(_imageHeight, _imageWidth),

        math.min(_imageHeight, _imageWidth),

        screen.height,

        screen.width,

        _model),

    ],

),

),

Container(

    child: Center(

        child: SizedBox.expand(

            child: FlatButton(

                highlightColor: Hexcolor('#A8DEE0'),

                splashColor: Hexcolor('#F9E2AE'),

                onPressed: () => getCapImage(),

```

```

        child: Text("Image Captioning",
          style: TextStyle(
            fontSize: 27.0,
            color: Colors.white,
            fontWeight: FontWeight.bold)))))),
    color: Hexcolor('6d597a')),
  Container(
    child: Center(
      child: SizedBox.expand(
        child: FlatButton(
          highlightColor: Hexcolor('#A8DEE0'),
          splashColor: Hexcolor('#F9E2AE'),
          onPressed: () => od.optionsDialogBox(context),
          child: Text("Text Extraction from Images",
            style: TextStyle(
              fontSize: 27.0,
              color: Colors.white,
              fontWeight: FontWeight.bold)))))),
        color: Hexcolor('b56576')),
  Container(

```

```

child: Center(
  child: SizedBox.expand(
    child: FlatButton(
      highlightColor: Hexcolor('#F9E2E'),
      splashColor: Hexcolor('#FBC78D'),
      onPressed: () => getCurrlImage(),
      child: Text("Currency Identifier",
        style: TextStyle(
          fontSize: 27.0,
          color: Colors.white,
          fontWeight: FontWeight.bold)))))),
  color: Hexcolor('e56b6f')),
Container(
  child: Center(
    child: SizedBox.expand(
      child: FlatButton(
        highlightColor: Colors.yellow[900],
        splashColor: Colors.yellow[500],
        onPressed: () => sd.sosDialogBox(context),
        child: Text("SOS Settings",

```

```

        style: TextStyle(
          fontSize: 27.0,
          color: Colors.white,
          fontWeight: FontWeight.bold))))),
      color: Hexcolor('eaac8b')),
    ],
    scrollDirection: Axis.horizontal,
    pageSnapping: true,
    physics: BouncingScrollPhysics(),
  ),
);
}

```

```

_speakPage(int a) async {
  if (a == 0) {
    await flutterTts.speak("Live object detection");
  } else if (a == 1) {
    if (await Vibration.hasVibrator()) {
      Vibration.vibrate(amplitude: 128, duration: 1000);
    }
  }
}

```

```

    await flutterTts.speak("Image Captioning");

} else if (a == 2) {

    if (await Vibration.hasVibrator()) {

        Vibration.vibrate(amplitude: 128, duration: 1400);

    }

    await flutterTts.speak("Text Extraction from images");

} else if (a == 3) {

    if (await Vibration.hasVibrator()) {

        Vibration.vibrate(amplitude: 128, duration: 1800);

    }

    await flutterTts.speak("Currency Identifier");

} else if (a == 4) {

    if (await Vibration.hasVibrator()) {

        Vibration.vibrate(amplitude: 128, duration: 2200);

    }

    await flutterTts.speak("SOS Settings");

}

}

}

```

iii) splashscreen.dart

```

import 'package:flutter/material.dart';

import 'package:splashscreen/splashscreen.dart';


import 'home.dart';

import 'main.dart';


class MySplash extends StatefulWidget {

  @override

  _MySplashState createState() => _MySplashState();

}


class _MySplashState extends State<MySplash> {

  @override

  Widget build(BuildContext context) {

    return SplashScreen(

      seconds: 3,

      navigateAfterSeconds: new HomePage(cameras),

      backgroundColor: Colors.white,

      photoSize: 150,

```

```

        loaderColor: Colors.blueAccent,

        image: Image.asset('assets/icon.jpg'),

    );

}

}

```

iv) currency.dart

```

import 'dart:io';

import 'package:tflite/tflite.dart';

import 'package:flutter/material.dart';

import 'package:hexcolor/hexcolor.dart';

import 'package:flutter_tts/flutter_tts.dart';

import 'package:vibration/vibration.dart';


import '../home.dart';

class CurrPage {

    static File currImage;

    static BuildContext context;

    static List _output;

```

```

static final FlutterTts flutterTts = FlutterTts();

static void currencyDetect(BuildContext buildContext, File img) {

  loadModel().then((value) {

    // setState(() {});

  });

  context = buildContext;

  currImage = img;

  speakCurrencyValue();

}

static classifyCurrency(File image) async {

  var output = await Tflite.runModelOnImage(

    path: image.path,

    numResults: 7,

    threshold: 0.5,

    imageMean: 127.5,

    imageStd: 127.5);

  print("This is the $output");

  print("This is the ${output[0]['label']}");

```

```

dynamic label = output[0]['label'];

print(label.runtimeType);

if (await Vibration.hasVibrator()) {

  Vibration.vibrate(amplitude: 500, duration: 200);

}

_speak(label);

_output = output;

showCaptionDialog(label, image);

}

```

```

static loadModel() async {

  await Tflite.loadModel(

    model: 'assets/cash_model_unquant.tflite',

    labels: 'assets/cash_labels.txt',

  );

}

```

```

static speakCurrencyValue() {

  classifyCurrency(currImage);

}

```

```

static Future _speak(String output) async {

  if (await Vibration.hasVibrator()) {

    Vibration.vibrate(amplitude: 500, duration: 200);

  }

  await flutterTts.speak(output);

}

```

```

static Future<void> showCaptionDialog(String text, File picture) async {

  showGeneralDialog(

    barrierColor: Colors.black.withOpacity(0.5),

    transitionBuilder: (context, a1, a2, widget) {

      return Transform.scale(

        scale: a1.value,

        child: Opacity(

          opacity: a1.value,

          child: AlertDialog(

            shape: OutlineInputBorder(

              borderRadius: BorderRadius.circular(16.0)),

            title: Text('Currency Identification'),

```

```

content: SingleChildScrollView(
  child: Column(
    mainAxisAlignment: MainAxisAlignment.spaceBetween,
    children: <Widget>[
      new Container(
        width: 300.0,
        height: 420.0,
        //decoration: BoxDecoration(borderRadius:
BorderRadius.all(Radius.circular(16.0))),
        child: RaisedButton(
          onPressed: () {
            _speak(text);
          },
          padding: const EdgeInsets.all(10.0),
          child: const Text(
            'Replay',
            style: TextStyle(
              color: Colors.white,
              fontWeight: FontWeight.bold),
          ),

```

```

        color: Hexcolor('e56b6f'),

        elevation: 5.0,

        shape: new RoundedRectangleBorder(

            borderRadius: new BorderRadius.circular(16.0)),

    ),

),

new Container(

    width: 300.0,

    height: 20,

),

new Image.file(picture),

    SizedBox(width: 20),

    new Text("$text"),

],

),

),

),

),

),

);

},

```

```

        transitionDuration: Duration(milliseconds: 200),

        barrierDismissible: true,

        barrierLabel: "",

        context: context,

        pageBuilder: (context, animation1, animation2) {});
}

```

```

static void _stopTts() {

    flutterTts.stop();

}

}

```

v) image_captioning.dart

```

import 'package:flutter/material.dart';

import 'dart:io';

import 'package:http/http.dart' as http;

import 'dart:convert';

import 'package:imgur/imgur.dart' as imgur;

import 'package:path/path.dart';

import 'package:flutter_tts/flutter_tts.dart';

import 'package:hexcolor/hexcolor.dart';

```

```
import 'package:vibration/vibration.dart';
```

```
class imgCap {
```

```
  File capImage;
```

```
  Context context;
```

```
  static final FlutterTts flutterTts = FlutterTts();
```

```
  static uploadImg(BuildContext context, File image) async {
```

```
    print("stage 1");
```

```
    final client =
```

```
      imgur.Imgur(imgur.Authentication.fromToken('INSERT YOUR IMGUR TOKEN  
HERE'));
```

```
    print("stage 2");
```

```
    /// Upload an image from path
```

```
    String imgLink = await client.image
```

```
      .uploadImage(
```

```
        imagePath: image.path,
```

```
      )
```

```

        .then((image) => image.link);

print("stage 3");

String caption = await extractCaption(imgLink);

print("stage 5");

_speak(caption);

showCaptionDialog(context, caption, image);
}

```

```

static _speak(String output) async {

  if (await Vibration.hasVibrator()) {

    Vibration.vibrate(amplitude: 500, duration: 200);

  }

  await flutterTts.speak(output);

}

```

```

static Future extractCaption(String imgLink) async {

  print("stage 4 ");

  var caption = await http.post('https://bow-flannel-food.glitch.me/caption',

    body: {"image_url": imgLink});

  print("stage a");

```

```

var parsed = json.decode(caption.body);

print("stage b");

return parsed.toString();
}

```

```

static Future<void> showCaptionDialog(
    BuildContext context, String text, File picture) async {
  showGeneralDialog(
    barrierColor: Colors.black.withOpacity(0.5),
    transitionBuilder: (context, a1, a2, widget) {
      return Transform.scale(
        scale: a1.value,
        child: Opacity(
          opacity: a1.value,
          child: AlertDialog(
            shape: OutlineInputBorder(
              borderRadius: BorderRadius.circular(16.0)),
            title: Text('Image Caption'),
            content: SingleChildScrollView(
              child: Column(

```

```

children: <Widget>[

  new Container(

    width: 300.0,

    height: 250,

    child: RaisedButton(

      onPressed: () {

        _speak(text);

      },

      padding: const EdgeInsets.all(10.0),

      child: const Text(

        'Replay',

        style: TextStyle(

          color: Colors.white,

          fontWeight: FontWeight.bold),

        ),

      color: Hexcolor('6d597a'),

      elevation: 5.0,

      shape: new RoundedRectangleBorder(

        borderRadius: new BorderRadius.circular(16.0)),

    ),

```

```

),

new Container(

  width: 300.0,

  height: 20,

),

new Container(

  width: 300.0,

  height: 250,

  child: RaisedButton(

    onPressed: _stopTts,

    padding: const EdgeInsets.all(10.0),

    child: const Text(

      'Stop',

      style: TextStyle(

        color: Colors.white,

        fontWeight: FontWeight.bold),

      ),

    color: Hexcolor('6d597a'),

    elevation: 5.0,

    shape: new RoundedRectangleBorder(

```

```

        borderRadius: new BorderRadius.circular(16.0)),
    ),
    new Container(
      width: 300.0,
      height: 20,
    ),
    new Image.file(picture),
    SizedBox(width: 20),
    new Text("$text"),
  ],
),
),
),
),
),
);
},
transitionDuration: Duration(milliseconds: 200),
barrierDismissible: true,
barrierLabel: "",
context: context,

```

```
        pageBuilder: (context, animation1, animation2) {});  
    }  
}
```

```
static Future _stopTts() async {  
    if (await Vibration.hasVibrator()) {  
        Vibration.vibrate(amplitude: 100, duration: 200);  
    }  
    flutterTts.stop();  
}  
}
```

vi) bndbox.dart

```
import 'dart:async';  
  
import 'package:flutter/material.dart';  
import 'package:flutter_tts/flutter_tts.dart';  
import 'dart:math' as math;  
import 'models.dart';  
  
class BndBox extends StatefulWidget {  
    final List<dynamic> results;
```

```
final int previewH;
```

```
final int previewW;
```

```
final double screenH;
```

```
final double screenW;
```

```
final String model;
```

```
BndBox(this.results, this.previewH, this.previewW, this.screenH, this.screenW,  
    this.model);
```

```
@override
```

```
_BndBox createState() => new _BndBox();
```

```
}
```

```
class _BndBox extends State<BndBox> {
```

```
    final FlutterTts flutterTts = FlutterTts();
```

```
    var _timer;
```

```
@override
```

```
void initState() {
```

```
    _timer = new Timer.periodic(
```

```

        Duration(milliseconds: 800), (Timer timer) => _speak());

    super.initState();
}

```

```

@override

void dispose() {

    _timer.cancel();

    super.dispose();

}

```

```

String speak;

String prevString;

```

```

@override

Widget build(BuildContext context) {

    List<Widget> _renderBoxes() {

        return widget.results.map((re) {

            var _x = re["rect"]["x"];

            var _w = re["rect"]["w"];

            var _y = re["rect"]["y"];

```

```

var _h = re["rect"]["h"];

var scaleW, scaleH, x, y, w, h;

if (widget.screenH / widget.screenW >
    widget.previewH / widget.previewW) {

    scaleW = widget.screenH / widget.previewH * widget.previewW;

    scaleH = widget.screenH;

    var difW = (scaleW - widget.screenW) / scaleW;

    x = (_x - difW / 2) * scaleW;

    w = _w * scaleW;

    if (_x < difW / 2) w -= (difW / 2 - _x) * scaleW;

    y = _y * scaleH;

    h = _h * scaleH;

} else {

    scaleH = widget.screenW / widget.previewW * widget.previewH;

    scaleW = widget.screenW;

    var difH = (scaleH - widget.screenH) / scaleH;

    x = _x * scaleW;

    w = _w * scaleW;

    y = (_y - difH / 2) * scaleH;

```

```

h = _h * scaleH;

if (_y < difH / 2) h -= (difH / 2 - _y) * scaleH;

}

```

```

speak = re["detectedClass"];

```

```

return Positioned(

  left: math.max(0, x),

  top: math.max(0, y),

  width: w,

  height: h,

  child: Container(

    padding: EdgeInsets.only(top: 5.0, left: 5.0),

    decoration: BoxDecoration(

      border: Border.all(

        color: Color.fromRGBO(37, 213, 253, 1.0),

        width: 3.0,

      ),

    ),

    child: Text(

```

```

        "${re["detectedClass"]} ${re["confidenceInClass"]} *
100).toStringAsFixed(0)}%",

        style: TextStyle(

            color: Color.fromRGBO(37, 213, 253, 1.0),

            fontSize: 14.0,

            fontWeight: FontWeight.bold,

        ),

    ),

),

);

}).toList();

}

```

```

List<Widget> _renderStrings() {

    double offset = -10;

    return widget.results.map((re) {

        offset = offset + 14;

        print(re["label"]);

        return Positioned(

            left: 10,

```

```

    top: offset,

    width: widget.screenW,

    height: widget.screenH,

    child: Text(

      "${re["label"]} ${re["confidence"] * 100}.toStringAsFixed(0)}%",

      style: TextStyle(

        color: Color.fromRGBO(37, 213, 253, 1.0),

        fontSize: 14.0,

        fontWeight: FontWeight.bold,

      ),

    ),

  );

}).toList();

}

return Stack(

  children: widget.model == mobilenet ? _renderStrings() : _renderBoxes(),

);

}

```

```

Future _speak() async {
    speak != null && speak != prevString ? await flutterTts.speak(speak) : null;

    prevString = speak;
}
}

```

vii) camera.dart

```

import 'package:flutter/material.dart';

import 'package:camera/camera.dart';

import 'package:tflite/tflite.dart';

import 'dart:math' as math;

import 'models.dart';

typedef void Callback(List<dynamic> list, int h, int w);

class Camera extends StatefulWidget {
    final List<CameraDescription> cameras;

    final Callback setRecognitions;

    final String model;

```

```
Camera(this.cameras, this.model, this.setRecognitions);
```

```
@override
```

```
_CameraState createState() => new _CameraState();
```

```
}
```

```
class _CameraState extends State<Camera> {
```

```
  CameraController controller;
```

```
  bool isDetecting = false;
```

```
@override
```

```
void initState() {
```

```
  super.initState();
```

```
  if (widget.cameras == null || widget.cameras.length < 1) {
```

```
    print('No camera is found');
```

```
  } else {
```

```
    controller = new CameraController(
```

```
      widget.cameras[0],
```

```
      ResolutionPreset.high,
```

```

);

controller.initialize().then((_) {

  if (!mounted) {

    return;

  }

  setState(() {});

  controller.startImageStream((CameraImage img) {

    if (!isDetecting) {

      isDetecting = true;

      int startTime = new DateTime.now().millisecondsSinceEpoch;

      if (widget.model == mobilenet) {

        Tflite.runModelOnFrame(

          bytesList: img.planes.map((plane) {

            return plane.bytes;

          }).toList(),

          imageHeight: img.height,

          imageWidth: img.width,

```

```

        numResults: 2,

    ).then((recognitions) {

        int endTime = new DateTime.now().millisecondsSinceEpoch;

        print("Detection took ${endTime - startTime}");

        widget.setRecognitions(recognitions, img.height, img.width);

        isDetecting = false;

    });

} else if (widget.model == posenet) {

    Tflite.runPoseNetOnFrame(

        bytesList: img.planes.map((plane) {

            return plane.bytes;

        }).toList(),

        imageHeight: img.height,

        imageWidth: img.width,

        numResults: 2,

    ).then((recognitions) {

        int endTime = new DateTime.now().millisecondsSinceEpoch;

        print("Detection took ${endTime - startTime}");

```

```

        widget.setRecognitions(recognitions, img.height, img.width);

        isDetecting = false;

    });

} else {

    Tflite.detectObjectOnFrame(

        bytesList: img.planes.map((plane) {

            return plane.bytes;

        }).toList(),

        model: widget.model == yolo ? "YOLO" : "SSDMobileNet",

        imageHeight: img.height,

        imageWidth: img.width,

        imageMean: widget.model == yolo ? 0 : 127.5,

        imageStd: widget.model == yolo ? 255.0 : 127.5,

        numResultsPerClass: 1,

        threshold: widget.model == yolo ? 0.2 : 0.4,

    ).then((recognitions) {

        int endTime = new DateTime.now().millisecondsSinceEpoch;

        print("Detection took ${endTime - startTime}");
    });
}

```

```
widget.setRecognitions(recognitions, img.height, img.width);
```

```
isDetecting = false;
```

```
});
```

```
}
```

```
}
```

```
});
```

```
});
```

```
}
```

```
}
```

```
@override
```

```
void dispose() {
```

```
  controller?.dispose();
```

```
  super.dispose();
```

```
}
```

```
@override
```

```
Widget build(BuildContext context) {
```

```

if (controller == null || !controller.value.isInitialized) {

    return Container();

}

var tmp = MediaQuery.of(context).size;

var screenH = math.max(tmp.height, tmp.width);

var screenW = math.min(tmp.height, tmp.width);

tmp = controller.value.previewSize;

var previewH = math.max(tmp.height, tmp.width);

var previewW = math.min(tmp.height, tmp.width);

var screenRatio = screenH / screenW;

var previewRatio = previewH / previewW;

return OverflowBox(

    maxHeight:

        screenRatio > previewRatio ? screenH : screenW / previewW * previewH,

    maxWidth:

        screenRatio > previewRatio ? screenH / previewH * previewW : screenW,

    child: CameraPreview(controller),

);

```

```
}  
  
}
```

viii) models.dart

```
const String mobilenet = "MobileNet";  
  
const String ssd = "SSD MobileNet";  
  
const String yolo = "Tiny YOLOv2";  
  
const String posenet = "PoseNet";
```

ix) dialog_ocr.dart

```
import 'package:flutter/cupertino.dart';  
  
import 'package:flutter/material.dart';  
  
import 'package:flutter_icons/flutter_icons.dart';  
  
import 'package:flutter_tts/flutter_tts.dart';  
  
import 'package:hexcolor/hexcolor.dart';  
  
import 'package:image_picker/image_picker.dart';  
  
import 'package:simple_ocr_plugin/simple_ocr_plugin.dart';  
  
import 'package:vibration/vibration.dart';  
  
  
class ocrDialog {  
  
  final FlutterTts flutterTts = FlutterTts();
```

```

Future<void> showOCRDialog(
    String text, PickedFile picture, BuildContext context) async {
    final pngByteData = await picture.readAsBytes();
    showGeneralDialog(
        barrierColor: Colors.black.withOpacity(0.5),
        transitionBuilder: (context, a1, a2, widget) {
            return Transform.scale(
                scale: a1.value,
                child: Opacity(
                    opacity: a1.value,
                    child: AlertDialog(
                        shape: OutlineInputBorder(
                            borderRadius: BorderRadius.circular(16.0)),
                        title: Text('Text Detected'),
                        content: SingleChildScrollView(
                            child: Column(
                                children: <Widget>[
                                    new Container(
                                        width: 300.0,
                                        height: 150,

```

```

child: RaisedButton(
  onPressed: _stopTts,
  padding: const EdgeInsets.all(10.0),
  child: const Text('Stop',
    style: TextStyle(
      color: Colors.white,
      fontWeight: FontWeight.bold)),
  color: Hexcolor('b56576'),
  elevation: 5.0,
  shape: new RoundedRectangleBorder(
    borderRadius: new BorderRadius.circular(16.0)),
),
),
new Container(height: 10),
new Container(
  width: 300.0,
  height: 150,
  child: RaisedButton(
    onPressed: () {
      _speakOCR(text);

```

```

    },
    padding: const EdgeInsets.all(10.0),
    child: const Text('Replay',
      style: TextStyle(
        color: Colors.white,
        fontWeight: FontWeight.bold)),
    color: Hexcolor('b56576'),
    elevation: 5.0,
    shape: new RoundedRectangleBorder(
      borderRadius: new BorderRadius.circular(16.0)),
  ),
),
new Container(height: 10),
new Container(
  width: 300.0,
  height: 150,
  child: RaisedButton(
    onPressed: _pauseTts,
    padding: const EdgeInsets.all(10.0),
    child: const Text('Pause',

```

```

        style: TextStyle(
          color: Colors.white,
          fontWeight: FontWeight.bold)),
      color: Hexcolor('b56576'),
      elevation: 5.0,
      shape: new RoundedRectangleBorder(
        borderRadius: new BorderRadius.circular(16.0)),
    ),
  ),
  new Container(height: 10),
  new Image.memory(pngByteData),
  SizedBox(width: 20),
  new Text("$text"),
],
),
),
),
),
),
);
},

```

```

        transitionDuration: Duration(milliseconds: 200),

        barrierDismissible: true,

        barrierLabel: "",

        context: context,

        pageBuilder: (context, animation1, animation2) {});
}

```

```

Future _stopTts() async {

    if (await Vibration.hasVibrator()) {

        Vibration.vibrate(amplitude: 100, duration: 200);

    }

    flutterTts.stop();

}

```

```

void _pauseTts() {

    flutterTts.pause();

}

```

```

Future _speakOCR(String text) async {

    if (await Vibration.hasVibrator()) {

```

```

    Vibration.vibrate(amplitude: 500, duration: 200);
  }

  await flutterTts.speak(text);
}

Future<void> optionsDialogBox(BuildContext context) {
  return showDialog(
    context: context,
    builder: (BuildContext context) {
      return AlertDialog(
        content: new SingleChildScrollView(
          child: new ListBody(
            children: <Widget>[
              RichText(
                text: TextSpan(
                  style: Theme.of(context).textTheme.bodyText2,
                  children: [
                    WidgetSpan(
                      child: Padding(
                        padding:

```

```

        const EdgeInsets.symmetric(horizontal: 3.0),
        child: Icon(FlutterIcons.photo_camera_mdi),
      ),
    ),
    TextSpan(
      text: 'Choose mode',
      style: new TextStyle(
        fontWeight: FontWeight.bold, fontSize: 20),
    ),
  ],
),
),
Padding(
  padding: EdgeInsets.all(8.0),
),
GestureDetector(
  child: new Text('Take a picture'),
  onTap: () {
    openCamera(context);
  },

```

```

    ),
    Padding(
      padding: EdgeInsets.all(8.0),
    ),
    GestureDetector(
      child: new Text('Select from gallery'),
      onTap: () {
        openGallery(context);
      },
    ),
  ],
),
),
);
});
}

```

```

Future<void> openCamera(BuildContext context) async {
  ImagePicker ip = new ImagePicker();
  var picture = await ip.getImage(

```

```

        source: ImageSource.camera,
    );

    var _extractText = await SimpleOcrPlugin.performOCR(image.path);
    print(_extractText.substring(20));

    _speakOCR(_extractText.substring(20, _extractText.length - 15));

    showOCRDialog(
        _extractText.substring(20, _extractText.length - 15), image, context);
}

```

```

Future<void> openGallery(BuildContext context) async {

    ImagePicker ip = new ImagePicker();

    var image = await ip.getImage(
        source: ImageSource.gallery,
    );

    var _extractText = await SimpleOcrPlugin.performOCR(image.path);
    print(_extractText.substring(20));

    _speakOCR(_extractText.substring(20, _extractText.length - 15));

    showOCRDialog(
        _extractText.substring(20, _extractText.length - 15), image, context);
}

```

```
}
```

x) sos_dialog.dart

```
import 'package:flutter/material.dart';
```

```
import 'package:hexcolor/hexcolor.dart';
```

```
import 'package:shared_preferences/shared_preferences.dart';
```

```
class sosDialog {
```

```
    TextEditingController _controller1 = new TextEditingController();
```

```
    TextEditingController _controller2 = new TextEditingController();
```

```
    TextEditingController _controller3 = new TextEditingController();
```

```
    TextEditingController _controller4 = new TextEditingController();
```

```
Future<void> sosDialogBox(BuildContext context) {
```

```
    return showDialog(
```

```
        context: context,
```

```
        builder: (BuildContext context) {
```

```
            return AlertDialog(
```

```
                title: Text("Enter phone numbers you would like to contact"),
```

```
                content: new SingleChildScrollView(
```

```

        child: new ListBody(children: <Widget>[
TextFormField(
    controller: _controller4,
    decoration: InputDecoration(
        labelText: 'Enter your name:',
    ),
),
TextFormField(
    keyboardType: TextInputType.number,
    controller: _controller1,
    decoration: InputDecoration(
        labelText: 'Enter phone number 1:',
    ),
),
TextFormField(
    keyboardType: TextInputType.number,
    controller: _controller2,
    decoration: InputDecoration(
        labelText: 'Enter phone number 2:',
    ),

```

```

),

TextFormField(

  keyboardType: TextInputType.number,

  controller: _controller3,

  decoration: InputDecoration(

    labelText: 'Enter phone number 3:',

  ),

),

new SizedBox(height: 10),

new RaisedButton(

  onPressed: () {

    setNumbers((_controller1.text), (_controller2.text),

      (_controller3.text), _controller4.text);

  },

  color: Hexcolor('eaac8b'),

  child: Text(

    "Save Information",

    style: TextStyle(

      color: Colors.white, fontWeight: FontWeight.bold),

  ),

```

```

        elevation: 5.0,

        shape: new RoundedRectangleBorder(

            borderRadius: new BorderRadius.circular(16.0)),

        )

    ]));

});

}

```

```

setNumbers(String n1, String n2, String n3, String n) async {

    SharedPreferences prefs = await SharedPreferences.getInstance();

    print(n1);

    print(n2);

    print(n3);

    await prefs.setString('n1', n1);

    await prefs.setString('n2', n2);

    await prefs.setString('n3', n3);

    await prefs.setString('name', n);

}

}

```

xi) app.py

```
from flask import Flask, request, jsonify
```

```
from captionbot import CaptionBot
```

```
app = Flask(__name__)
```

```
@app.route('/', methods = ['POST', 'GET'])
```

```
def home():
```

```
    return "This is the Home Endpoint"
```

```
@app.route('/caption', methods = ['POST'])
```

```
def caption():
```

```
    image_caption_result = "
```

```
    if request.method == 'POST':
```

```
        image_url = request.form['image_url']
```

```
        image_caption_generator = CaptionBot()
```

```
        image_caption_result =
```

```
        jsonify(image_caption_generator.url_caption(str(image_url)))
```

```
        return image_caption_result
```

```
if __name__ == '__main__':
```

```
app.run(debug=False)
```

xii) Dockerfile

```
FROM python:alpine3.7
```

```
COPY . /app
```

```
WORKDIR /app
```

```
RUN pip install -r requirements.txt
```

```
EXPOSE 5000
```

```
ENTRYPOINT [ "python" ]
```

```
CMD [ "app.py" ]
```

xiii) caption_generation.py

```
import pickle
```

```
from keras.models import load_model
```

```
from keras.preprocessing import image
```

```
import numpy as np
```

```
from keras.applications.resnet50 import preprocess_input
```

```
from keras.preprocessing.sequence import pad_sequences
```

```
MAX_LENGTH = 40
```

```
new_model = load_model('models/image-captioning.h5')
```

```
new_model._make_predict_function()
```

```

Rmodel = load_model('models/pretrained-resnet.h5')

Rmodel._make_predict_function()


w2i_file = open('models/words-indices.p', 'rb')

words_to_indices2 = pickle.load(w2i_file)


i2w_file = open('models/indices-words.p', 'rb')

indices_to_words2 = pickle.load(i2w_file)


def get_enc(img_path):

    img = image.load_img(img_path, target_size=(224, 224))

    x = image.img_to_array(img)

    x = np.expand_dims(x, axis=0)

    x = preprocess_input(x)

    features = Rmodel.predict(x)

    photo = features.squeeze()

    return photo


def get_caption(img_path):

    enc = get_enc(img_path)

```

```
caption = greedy_search(enc)
```

```
return caption
```

```
def greedy_search(photo):
```

```
    photo = photo.reshape(1, 2048)
```

```
    in_text = '<start>'
```

```
    for i in range(MAX_LENGTH):
```

```
        sequence = [words_to_indices2[s] for s in in_text.split(' ')]
```

```
            if s in words_to_indices2]
```

```
        sequence = pad_sequences([sequence], maxlen=MAX_LENGTH,
```

```
                                padding='post')
```

```
        y_pred = new_model.predict([photo, sequence], verbose=0)
```

```
        y_pred = np.argmax(y_pred[0])
```

```
        word = indices_to_words2[y_pred]
```

```
        in_text += ' ' + word
```

```
        if word == '<end>':
```

```
            break
```

```
    final_text = in_text.split()
```

```
    final_text = final_text[1:-1]
```

```
    final_text = ' '.join(final_text)
```

```
return final_text
```

```
def beam_search2(photo, k):  
  
    photo = photo.reshape(1, 2048)  
  
    in_text = '<start>  
  
    sequence = [words_to_indices2[s] for s in in_text.split(' ') if s  
                in words_to_indices2]  
  
    sequence = pad_sequences([sequence], maxlen=MAX_LENGTH,  
                             padding='post')  
  
    y_pred = new_model.predict([photo, sequence], verbose=0)  
  
    predicted = []  
  
    y_pred = y_pred.reshape(-1)  
  
    for i in range(y_pred.shape[0]):  
        predicted.append((i, y_pred[i]))  
  
    predicted = sorted(predicted, key=lambda x: x[1])[:-1]  
  
    b_search = []  
  
    for i in range(k):  
        word = indices_to_words2[predicted[i][0]]  
  
        b_search.append((in_text + ' ' + word, predicted[i][1]))
```

```

for idx in range(MAX_LENGTH):

    b_search_square = []

    for text in b_search:

        if text[0].split(' ')[-1] == '<end>':

            break

        sequence = [words_to_indices2[s] for s in text[0].split(' '

            ) if s in words_to_indices2]

        sequence = pad_sequences([sequence], maxlen=MAX_LENGTH,

            padding='post')

        y_pred = new_model.predict([photo, sequence], verbose=0)

        predicted = []

        y_pred = y_pred.reshape(-1)

        for i in range(y_pred.shape[0]):

            predicted.append((i, y_pred[i]))

        predicted = sorted(predicted, key=lambda x: x[1])[::-1]

        for i in range(k):

            word = indices_to_words2[predicted[i][0]]

            b_search_square.append((text[0] + ' ' + word,

                predicted[i][1] * text[1]))

```

```

if len(b_search_square) > 0:

    b_search = (sorted(b_search_square, key=lambda x: x[1])[:-1])[:5]

final_text = b_search[0][0].split()

final_text = final_text[1:-1]

final_text = ' '.join(final_text)

return final_text

```

PLAGIARISM REPORT

Untitled document

Flask

Flask is a Micro Web-Framework that is used for developing Backend Interface for your Application. It is a lightweight framework which has a small footprint and can be used to set up things pretty quickly, with a simple yet extensible core.

Flask was created by Armin Ronacher of Pocoo, an international group of Python enthusiasts formed in 2004. It is designed to help getting started quick and easy, with the ability to scale up to complex applications. It began as a simple wrapper around Werkzeug and Jinja and has become one of the most popular Python web application frameworks.

Flask is currently being used to develop RESTful APIs that has got multiple endpoints and is seen as a fast and reliable way to deploy Machine Learning Models. Some of its features include:

- **Easy to Understand:** Flask is very easy to understand and to test. Using Flask, we can setup routes for our Backend Service as easy as writing Python functions.
- **Third-Party Modules:** Flask supports many Third-Party Modules which enhances the modularity of the Code and allows the Developers to scale it as they like.

10,396 words

Plagiarism

Back to all suggestions

Looks like your text is 100% original.
We found no matching text in our databases or on the Internet.

HIDE ASSISTANT

90
Overall score
See performance

Goals
4 of 4 set

All suggestions

Correctness
221 alerts

Clarity
Clear

Engagement
Very engaging

Delivery
Slightly off

Style guide
All good

Plagiarism