JHARKHAND
Rai University
UGC RECOGNISED UNIVERSITY
ACCREDITED BY NAAC

Olchiki OCR : An Android Application

By

Sunil Mardi
BC/18/005

Under Guidance
Of
Prof. Anuradha Sharma

Submitted to Department of
Computer Science and Information Technology

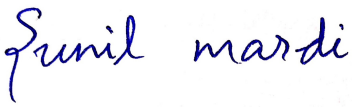
in partial fulfillment of the
requirements for the award of the degree

Bachelor of Computer Applications(BCA)
2021

CERTIFICATE OF ORIGINALITY

This is to certify that the project entitled **Olchiki OCR : An Android Application** submitted to **Jharkhand Rai University** in partial fulfillment of the requirement for the award of the degree of BACHELOR OF COMPUTER APPLICATIONS (BCA), is an authentic and original work carried out by **Mr. Sunil Mardi** with enrollment no. **BC/18/005** under my guidance.

The matter embodied in this project is genuine work done by the student and has not been submitted whether to this University or to any other University for the fulfillment of the requirements of any course of study.



……
Signature of the Student
Date: 25 June 2021

Sunil Mardi
Piska More, Ranchi

Enrollment No. BC/18/005

……
Signature of the
Guide/Coordinator
Date: 25 June 2021

Prof. Anuradha Sharma
Ranchi

……
Signature of the External
Date: 25 June 2021

Teacher
Place

ACKNOWLEDGEMENT

It gives me immense pleasure to express my deepest sense of gratitude and sincere thanks to my highly respected and esteemed guide **Prof. Anuradha Sharma, Computer Science and Information Technology, Jharkhand Rai University, Ranchi,** for her valuable guidance, encouragement and help for completing this work. Her useful suggestions for this whole work and co-operative behavior are sincerely acknowledged. I would like to express my sincere thanks to **Prof. (Dr.) Piyush Ranjan, *Dean (Academics), JRU*, *Ranchi*** for giving me this opportunity to undertake this project. I also wish to express my gratitude to **Prof. Anuradha Sharma, Coordinator (Computer Science & Information Technology), JRU, Ranchi** for her kind hearted support. I am also grateful to my faculty members for their constant support and guidance. I also wish to express my indebtedness to my parents as well as my family member whose blessings and support always helped me to face the challenges ahead. At the end I would like to express my sincere thanks to all my friends and others who helped me directly or indirectly during this project work.

Place: Ranchi
Date: 21 June 2021

Sunil Mardi
BC/18/005

CONTENT

Chapter 1

Introduction

Olchiki OCR

Olchiki OCR is an android application that recognizes the handwritten santhali language letters and converts them into editable text.This Application uses Machine Learning Techniques to recognize the letters. This application is developed using TensorFlow and Android Studio.All the letters of Santhali Script are included in the application. The name **Olchiki OCR** of the android application is based on the Santhali script name which is Ol Chiki.

App Logo

Ol Chiki

Ol Chiki is alphabetic, and does not share any of the syllabic properties of the other Indic scripts. It contains 30 letters and five basic diacritics. It has 6 basic vowels and additional three vowels are generated using **Gahla Tudag**. On commenting about Ol Chiki, Norman Zide (1996) observes, **"The shapes of the letters are not arbitrary, but reflect the names for the letters, which are words, usually the names of objects or actions representing conventionalized form in the pictorial shape of the characters."**

Pandit Raghunath Murmu is the inventor of Ol Chiki script. He was born in a village, called (Dahardih)Dandbose, on **5th May 1905** on the day of full moon in the district of Mayurbhanj, Orissa. After a brief stint in technical profession, he took up the job of teaching in Badomtolia high school. During this time, his interest was drawn into Santali literature. Santali is a language with its own special characteristics, and has a literature which dates back to the beginning of the 15th century. Naturally, he felt that Santals with their rich cultural heritage and tradition also need a separate script to preserve and promote their language, and therefore, he took up the work of inventing Ol Chiki script for writing Santali. The epoch making invention of Ol Chiki script was unveiled in 1925. In the novel **Bidu Chandan**, he has vividly described how god Bidu, and goddess Chandan who appear on Earth as human being would have naturally invented the Ol Chiki script in order to communicate with each other using written Santali. He wrote over 150 books covering a wide spectrum of subjects such as grammar, novels, drama, poetry, and story in Santali using Ol Chiki as a part of his extensive programme for culturally upgrading the Santhal community. "**Darege Dhan",** "**Sidhu-Kanhu"**, "**Bidu Chandan**" and "**Kherwal Bir**" are among the most acclaimed of his works. Pandit Raghunath Murmu is popularly known as **GURU**

GOMKE among the Santals, a title that was conferred on him by the Mayurbhanj Adibasi Mahasabha. Besides the Govt of West Bengal and Orissa, several other organizations/associations including Orissa Sahitya Academy have honored him in various ways and **Hon D. Litt.** was conferred on him by Ranchi University. The great thinker, philosopher, writer, and dramatist breathed his last breath on 1st February, 1982.

Pandit Raghunath Murmu

Need Ol Chiki script

In earlier times, all Santali writings were in Bengali, Devanagari, or Roman script. Although there have been impressive number of works by foreigner and non-Santal writers on dictionary, grammar, collection of folklore etc., these works are mostly intended for research purposes. Roman script was in extensive use for writing Santali and several books in Santali have been published using Roman script. But most of the creative literature were written by the native speakers in Bengali or Devanagari script. The use of different scripts for writing Santali has hindered the development and utilization of Santali language. This, in turn, has effectively marred the progress of Santali language in several fields such as philosophy, history, religion, science, novel, prose, poetry etc. The problem of using different scripts for the same language necessitated the invention of a new script for Santali, and it finally led to the invention of **Ol Chiki** by **Pandit Raghunath Murmu**.

After the invention of Ol Chiki, a large number of books have been written by various authors in Santali using Ol Chiki script. Types of books include (i) novels and short stories, (ii) poetries, songs, and religious sermons, (iii) books on Santal society, (iv) primary books for learning Ol Chiki, (v) books for learning primary mathematics, (vi) books on Santali grammars and related topics, and (vii) books on great tribal persons. Santali magazines in Ol Chiki are also being published regularly.

This script is also known as **Ol Cemet**, **Ol script**, **Ol Chiki Script** and also **Ol**. In Santali, **Ol** means writing and **Cemet** means learning . So, Ol Cemet' means the learning of writing.

The Ol Chiki letters are arranged in a matrix of 6 by 5, in which the six letters in the **first column of the matrix are vowels**, and the rest 24 letters are consonants. However, the five letters of the third column represent dual consonants, and this, eventually, helps to represent 29 consonants with the help of diacritic **Ahad**. Ol Chiki gives 5 basic diacritics, and the combination of diacritics **Mu Tudag and Gahla Tudag** gives rise to another diacritic, called **Mu-Gahla Tudag**. The matrix of Ol Chiki letters are listed with transliteration of alphabets, with pronunciation in brackets and their sounds in bracelets.

Letters

The values of the letters are as follows:

Letter	Name	IPA	Transliteration					Shape
			ALA-LC	Zide	Deva.	Beng.	Odia	
ᱚ	la	/ɔ/	a	o̤	अ	অ	ଅ	burning fire
ᱛ	at	/t/	t	t	त	ত	ତ	the Earth
ᱜ	ag	/k'/, /g/	g	k'	ग	গ	ଗ	vomiting mouth which produces the same sound as the name of the letter
ᱝ	ang	/ŋ/	ṃ	ṅ	◌̇	◌ং	◌ଂ	blowing air
ᱞ	al	/l/	l	l	ल	ল	ଲ	writing
ᱟ	laa	/a/	ā	a	आ	আ	ଆ	working in the field with a spade
ᱠ	aak	/k/	k	k	क	ক	କ	bird (sound of a swan)
ᱡ	aaj	/c'/, /j/	j	c'	ज	জ	ଜ	person pointing towards a third person with the right hand (saying he)
ᱢ	aam	/m/	m	m	म	ম	ମ	person pointing towards a second person with the left hand (saying you)
ᱣ	aaw	/w/, /v/	w	w	व	ওয়	ୱ	opening lips
ᱤ	li	/i/	i	i	इ	ই	ଇ	bending tree
ᱥ	is	/s/	s	s	स	স	ସ	plow
ᱦ	ih	/ʔ/, /h/	h̲	h	ह	হ	ହ	hands up
ᱧ	iny	/ɲ/	ñ	ñ	ञ	ঞ	ଞ	person pointing towards himself/herself with the

								left hand
ᱨ	ir	/r/	r	r	র	ৰ	ৱ্	sickle used for cutting or reaping
ᱩ	lu	/u/	u	u	উ	ঊ	ঊ	vessel used for preparing food
ᱪ	uch	/c/	c	c	চ	চ	চ্	peak of a mountain which is usually high
ᱫ	ud	/t’/, /d/	d	t’	দ	দ	দ্	mushroom
ᱱ	unn	/ɳ/	ṇ	ṇ	ণ	ণ	ণ্	picture of a flying bee (which makes this sound)
ᱬ	uy	/j/	y	y	য	য়	য়্	a man bending towards ground to cut something
ᱮ	le	/e/	e	e	এ	এ	এ	overflowing rivers changing course
ᱯ	ep	/p/	p	p	প	প	প্	person receiving with both hands
ᱰ	edd	/ɖ/	ḍ	ḍ	ড	ড	ড্	a man with two legs stretching towards his chest and mouth
ᱱ	en	/n/	n	n	ন	ন	ন্	thrashing grains with two legs
ᱵ	err	/ɽ/	ṛ	ṛ	ড়	ড়	ড়্	a path that turns to avoid an obstruction or a danger
ᱳ	lo	/o/	o	o	ও	ও	ও	a mouth when sounding this letter
ᱴ	ott	/ʈ/	ṭ	ṭ	ট	ট	ট্	camel hump
ᱦ	ob	/p’/, /b/	b	p’	ব	ব	ব্	curly hair
ᱶ	ov	/w̃/	ṅ	w̃	ঙ	ঙ	ঙ্	nasalized
ᱷ	oh	/ʰ/	h	(C)h	হ	হ	হ্	a man throwing something with one hand

Digits

Ol Chiki uses decimal system, and the symbol for basic digits 0-9 are as follows:

᱐	᱑	᱒	᱓	᱔	᱕	᱖	᱗	᱘	᱙
0	1	2	3	4	5	6	7	8	9

Objective of the Project

The objective of the project is to develop an android application which will extract text from an image file. The image file is an handwritten letter, word or sentence in Ol Chiki Script which is an script of Santhali Language. Santhali language has 30 letters.

Purpose

The purpose of this project is have a OCR which works on Ol Chiki Script. Different Languages have their own OCR and trained datasets of the language script.
Google's Cloud Vision supports so many different regional and international languages and their script but not the Santhali script . To have its place on the digital world, this project is being developed.

Scope

The scope of this project **Ol Chiki OCR** is to provide an efficient and enhanced software tool for the users to perform Document Image Analysis, document processing by reading and recognizing the characters in research, academic, governmental and business organizations that are having large pool of documented, scanned images in **Ol Chiki script** of **Santhali Language**.

Applications

This project can be applicable in several fields. Many examinations are conducted in Ol Chiki Script.West Bengal class 12th question papers were also published in Santhali, OL Chiki language. Santhali is written in Ol Chiki script and spoken by 6.4 million people, according to the 2001 census, in India alone and its speakers live mostly in Jharkhand, West Bengal, Odisha and Assam. It is also spoken in Bangladesh and Nepal.Santhali has become the first Indian tribal language to get a Wikipedia edition in its own script.

Santhali Literature or Santali is an optional subject in the UPSC Civil Services examination. Candidates who have studied Santhali in school and college or have learnt it as their native language should opt for this subject. All answers are required to be written in the **Ol Chiki script**.

Examination papers can be scanned and evaluated using OCR.

Chapter 2

SURVEY OF TECHNOLOGIES

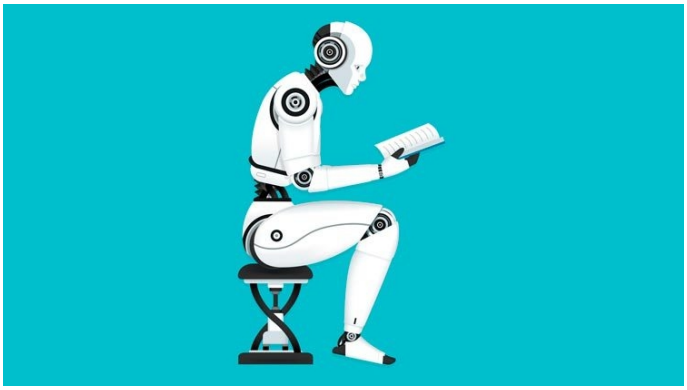
Machine Learning

Machine learning (**ML**) is the study of computer algorithms that improve automatically through experience and by the use of data. It is seen as a part of artificial intelligence. Machine learning algorithms build a model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to do so. Machine learning algorithms are used in a wide variety of applications, such as in medicine, email filtering, speech recognition, and computer vision, where it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks.

A subset of machine learning is closely related to computational statistics, which focuses on making predictions using computers; but not all machine learning is statistical learning. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a related field of study, focusing on exploratory data analysis through unsupervised learning. In its application across business problems, machine learning is also referred to as predictive analytics.

Machine learning involves computers discovering how they can perform tasks without being explicitly programmed to do so. It involves computers learning from data provided so that they carry out certain tasks. For simple tasks assigned to computers, it is possible to program algorithms telling the machine how to execute all steps required to solve the problem at hand; on the computer's part, no learning is needed. For more advanced tasks, it can be challenging for a human to manually create the needed algorithms. In practice, it can turn out to be more effective to help the machine develop its own algorithm, rather than having human programmers specify every needed step.

The discipline of machine learning employs various approaches to teach computers to accomplish tasks where no fully satisfactory algorithm is available. In cases where vast numbers of potential answers exist, one approach is to label some of the correct answers as valid. This can then be used as training data for the computer to improve the algorithm(s) it uses to determine correct answers. For example, to train a system for the task of digital character recognition, the MNIST dataset of handwritten digits has often been used.



Artificial intelligence

As a scientific endeavor, machine learning grew out of the quest for artificial intelligence. In the early days of AI as an academic discipline, some researchers were interested in having machines learn from data. They attempted to approach the problem with various symbolic methods, as well as what was then termed "neural networks"; these were mostly perceptrons and other models that were later found to be reinventions of the generalized linear models of statistics. Probabilistic reasoning was also employed, especially in automated medical diagnosis.

However, an increasing emphasis on the logical, knowledge-based approach caused a rift between AI and machine learning. Probabilistic systems were plagued by theoretical and practical problems of data acquisition and representation. By 1980, expert systems had come to dominate AI, and statistics was out of favor. Work on symbolic/knowledge-based learning did continue within AI, leading to inductive logic programming, but the more statistical line of research was now outside the field of AI proper, in pattern recognition and information retrieval. Neural networks research had been abandoned by AI and computer science around the same time. This line, too, was continued outside the AI/CS field, as "connectionism", by researchers from other disciplines including Hopfield, Rumelhart and Hinton. Their main success came in the mid-1980s with the reinvention of backpropagation.

Machine learning (ML), reorganized as a separate field, started to flourish in the 1990s. The field changed its goal from achieving artificial intelligence to tackling solvable problems of a practical nature. It shifted focus away from the symbolic approaches it had inherited from AI, and toward methods and models borrowed from statistics and probability theory

Data mining

Machine learning and data mining often employ the same methods and overlap significantly, but while machine learning focuses on prediction, based on *known* properties learned from the training data, data mining focuses on the discovery of (previously) *unknown* properties in the data (this is the analysis step of knowledge discovery in databases). Data mining uses many machine learning methods, but with different goals; on the other hand, machine learning also employs data mining methods as "unsupervised learning" or as a preprocessing step to improve learner accuracy. Much of the confusion between these two research communities (which do often have separate conferences and separate journals, ECML PKDD being a major exception) comes from the basic assumptions they work with: in machine learning, performance is usually evaluated with respect to the ability to *reproduce known* knowledge, while in knowledge discovery and data mining (KDD) the key task is the discovery of previously *unknown* knowledge. Evaluated with respect to known knowledge, an uninformed (unsupervised) method will easily be outperformed by other supervised methods, while in a typical KDD task, supervised methods cannot be used due to the unavailability of training data.

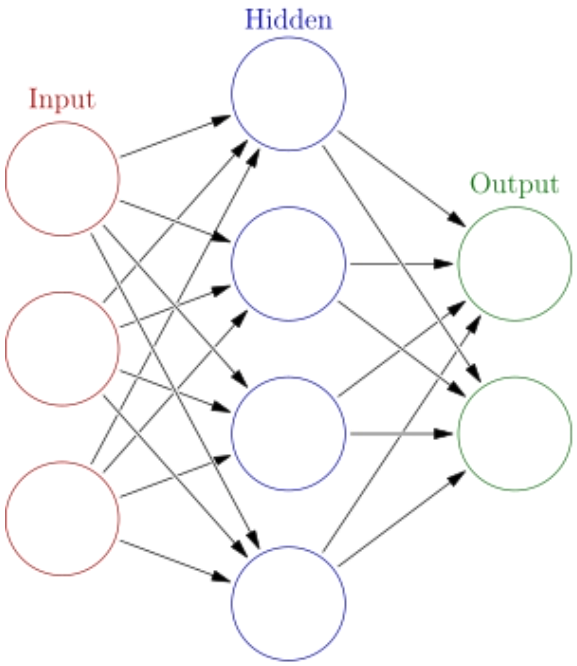
Approaches

Machine learning approaches are traditionally divided into three broad categories, depending on the nature of the "signal" or "feedback" available to the learning system:

- **Supervised learning**: The computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs.
- **Unsupervised learning**: No labels are given to the learning algorithm, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning).
- **Reinforcement learning**: A computer program interacts with a dynamic environment in which it must perform a certain goal (such as driving a vehicle or playing a game against an opponent). As it navigates its problem space, the program is provided feedback that's analogous to rewards, which it tries to maximize.

Models

Performing machine learning involves creating a model, which is trained on some training data and then can process additional data to make predictions. Various types of models have been used and researched for machine learning systems.



An artificial neural network is an interconnected group of nodes, akin to the vast network of neurons in a brain. Here, each circular node represents an artificial neuron and an arrow represents a connection from the output of one artificial

Applications

There are many applications for machine learning, including:

- Agriculture
- Anatomy
- Adaptive websites
- Affective computing
- Banking
- Bioinformatics
- Brain–machine interfaces
- Cheminformatics
- Citizen science
- Computer networks
- Computer vision
- Credit-card fraud detection
- Data quality
- DNA sequence classification
- Economics
- Financial market analysis
- General game playing
- Handwriting recognition
- Information retrieval
- Insurance
- Internet fraud detection
- Knowledge graph embedding
- Linguistics
- Machine learning control
- Machine perception
- Machine translation
- Marketing
- Medical diagnosis
- Natural language processing
- Natural language understanding
- Online advertising
- Optimization
- Recommender systems
- Robot locomotion
- Search engines
- Sentiment analysis
- Sequence mining

- Software engineering
- Speech recognition
- Structural health monitoring
- Syntactic pattern recognition
- Telecommunication
- Theorem proving
- Time series forecasting
- User behavior analytics
- Behaviorism

Softwares

Software suites containing a variety of machine learning algorithms include the following :

Free and open-source software

- Caffe
- CNTK
- Deeplearning4j
- DeepSpeed
- Infer.NET
- Keras
- LightGBM
- Mahout
- Mallet
- ML.NET
- mlpack
- MXNet
- Neural Lab
- OpenNN
- pandas (software)
- PyTorch
- ROOT (TMVA with ROOT)
- scikit-learn
- Shogun
- Spark MLlib
- SystemML
- TensorFlow
- Torch / PyTorch
- Weka / MOA
- Yooreeka

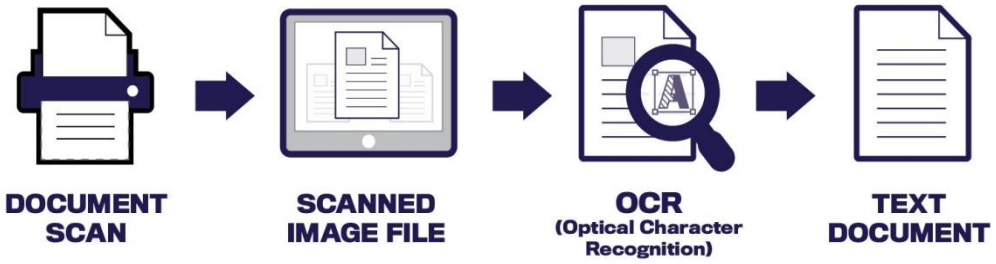
OCR

Optical Character Recognition, or OCR, is a technology that enables you to convert different types of documents, such as scanned paper documents, PDF files or images captured by a digital camera into editable and searchable data.

Imagine you've got a paper document - for example, magazine article, brochure, or PDF contract your partner sent to you by email. Obviously, a scanner is not enough to make this information available for editing, say in Microsoft Word. All a scanner can do is create an image or a snapshot of the document that is nothing more than a collection of black and white or color dots, known as a raster image. In order to extract and repurpose data from scanned documents, camera images or image-only PDF, you need an OCR software that would single out letters on the image, put them into words and then, words into sentences, thus enabling you to access and edit the content of the original document.

Recognition of Digital Camera Images

Images captured by a digital camera differ from scanned documents or image-only PDF. They often have defects such as distortion at the edges and dimmed light, making it difficult for most OCR applications to correctly recognize the text. It offers a range of features to improve the quality of such images, providing you with the ability to fully use the capabilities of your digital devices



TensorFlow

TensorFlow is a free and open-source software library for machine learning. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.

Tensorflow is a symbolic math library based on dataflow and differentiable programming. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache License 2.0 in 2015.

TensorFlow is Google Brain's second-generation system. Version 1.0.0 was released on February 11, 2017.While the reference implementation runs on single devices, TensorFlow can run on multiple CPUs and GPUs (with optional CUDA and SYCL extensions for general-purpose computing on graphics processing units). TensorFlow is available on 64-bit Linux, macOS, Windows, and mobile computing platforms including Android and iOS.

Its flexible architecture allows for the easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.

TensorFlow computations are expressed as stateful dataflow graphs. The name TensorFlow derives from the operations that such neural networks perform on multidimensional data arrays, which are referred to as *tensors*. During the Google I/O Conference in June 2016, Jeff Dean stated that 1,500 repositories on GitHub mentioned TensorFlow, of which only 5 were from Google.

In December 2017, developers from Google, Cisco, RedHat, CoreOS, and CaiCloud introduced Kubeflow at a conference. Kubeflow allows operation and deployment of TensorFlow on Kubernetes.

In March 2018, Google announced TensorFlow.js version 1.0 for machine learning in JavaScript.

In Jan 2019, Google announced TensorFlow 2.0.[18] It became officially available in Sep 2019.

In May 2019, Google announced TensorFlow Graphics for deep learning in computer graphics.

Android

Android is a mobile operating system based on a modified version of the Linux kernel and other open source software, designed primarily for touchscreen mobile devices such as smartphones and tablets. Android is developed by a consortium of developers known as the Open Handset Alliance and commercially sponsored by Google. It was unveiled in November 2007, with the first commercial Android device, the HTC Dream, being launched in September 2008.

It is free and open-source software; its source code is known as Android Open Source Project (AOSP), which is primarily licensed under the Apache License. However most Android devices ship with additional proprietary software pre-installed, most notably Google Mobile Services (GMS) which includes core apps such as Google Chrome, the digital distribution platform Google Play and associated Google Play Services development platform.

About 70 percent of Android smartphones run Google's ecosystem; some with vendor-customized user interface and software suite, such as *TouchWiz* and later *One UI* by Samsung, and *HTC Sense*. Competing Android ecosystems and forks include Fire OS (developed by Amazon) or LineageOS. However the "Android" name and logo are trademarks of Google which impose standards to restrict "uncertified" devices outside their ecosystem to use Android branding.

The source code has been used to develop variants of Android on a range of other electronics, such as game consoles, digital cameras, portable media players, PCs and others, each with a specialized user interface. Some well known derivatives include Android TV for televisions and Wear OS for wearable, both developed by Google. Software packages on Android, which use the APK format, are generally distributed through proprietary application stores like Google Play Store, Samsung Galaxy Store, Huawei AppGallery, Cafe Bazaar, and GetJar, or open source platforms like Aptoide or F-Droid.

Android has been the best-selling OS worldwide on smartphones since 2011 and on tablets since 2013. As of May 2021, it has over three billion monthly active users, the largest installed base of any operating system, and as of January 2021, the Google Play Store features over 3 million apps. The current stable version is Android 11, released on September 8, 2020.

Android Studio

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS and Linux based operating systems or as a subscription-based service in 2020. It is a replacement for the Eclipse Android Development Tools (E-ADT) as the primary IDE for native Android application development.

Android Studio was announced on May 16, 2013 at the Google I/O conference. It was in early access preview stage starting from version 0.1 in May 2013, then entered beta stage starting from version 0.8 which was released in June 2014. The first stable build was released in December 2014, starting from version 1.0.

Features

A specific feature of the Android Studio is an absence of the possibility to switch autosave feature off.

The following features are provided in the current stable version:

- Gradle-based build support
- Android-specific refactoring and quick fixes
- Lint tools to catch performance, usability, version compatibility and other problems
- ProGuard integration and app-signing capabilities
- Template-based wizards to create common Android designs and components
- A rich layout editor that allows users to drag-and-drop UI components, option to preview layouts on multiple screen configurations
- Support for building Android Wear apps
- Built-in support for Google Cloud Platform, enabling integration with Firebase Cloud Messaging (Earlier 'Google Cloud Messaging') and Google App Engine
- Android Virtual Device (Emulator) to run and debug apps in the Android studio.

Once an app has been compiled with Android Studio, it can be published on the Google Play Store. The application has to be in line with the Google Play Store developer content policy.

Chapter 3

Requirement and Analysis

Problem Definition

India is a country with variety of languages spoken.With the development of technology,we are working with international as well as regional languages.The present OCR system deals with English and few regional languages. Currently there is no OCR or any such technology for **Santhali** language or its script **Ol Chiki.** Santhali received global recognition when it got a Wikipedia edition in its own script. There was no OCR for Ol Chiki earlier. To have an OCR of Ol Chiki script this project is being developed.

Software & Hardware Requirement

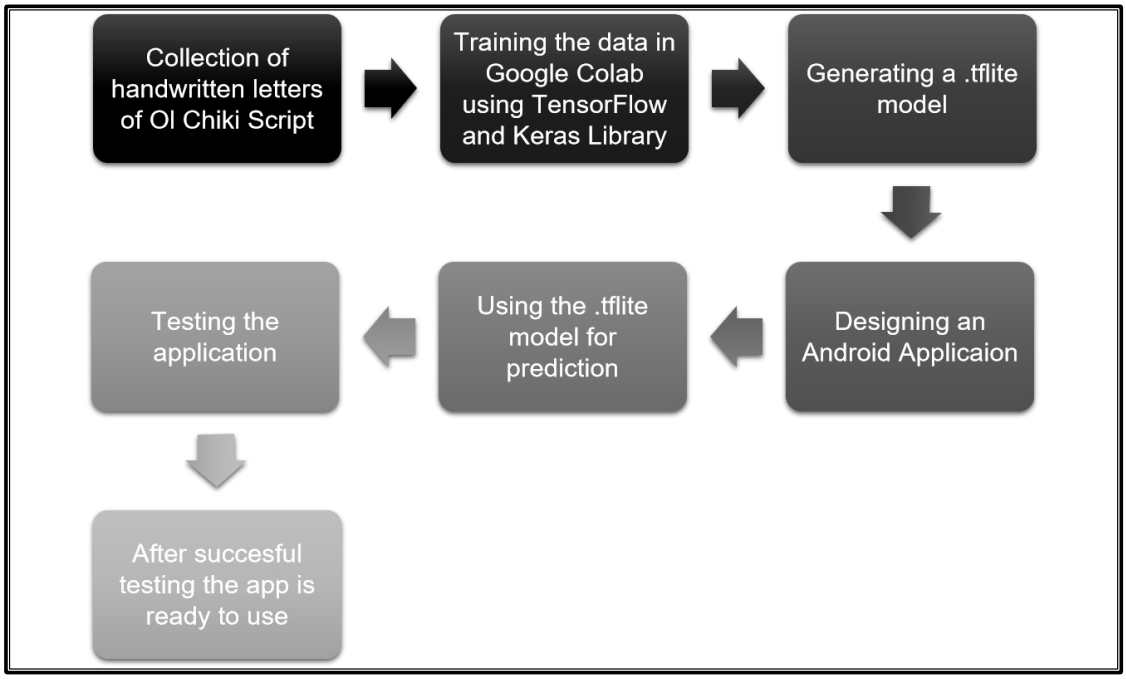
Google Colab	: for training the model with the letters of **Ol Chiki** script using **TensorFlow** and **Keras** library
Google Drive	: for storing dataset for training
Android Studio	:for developing an **Android Application** using **Kotlin** and **XML**
A Smartphone	: for testing

Chapter 4

System Design Implementation and Testing

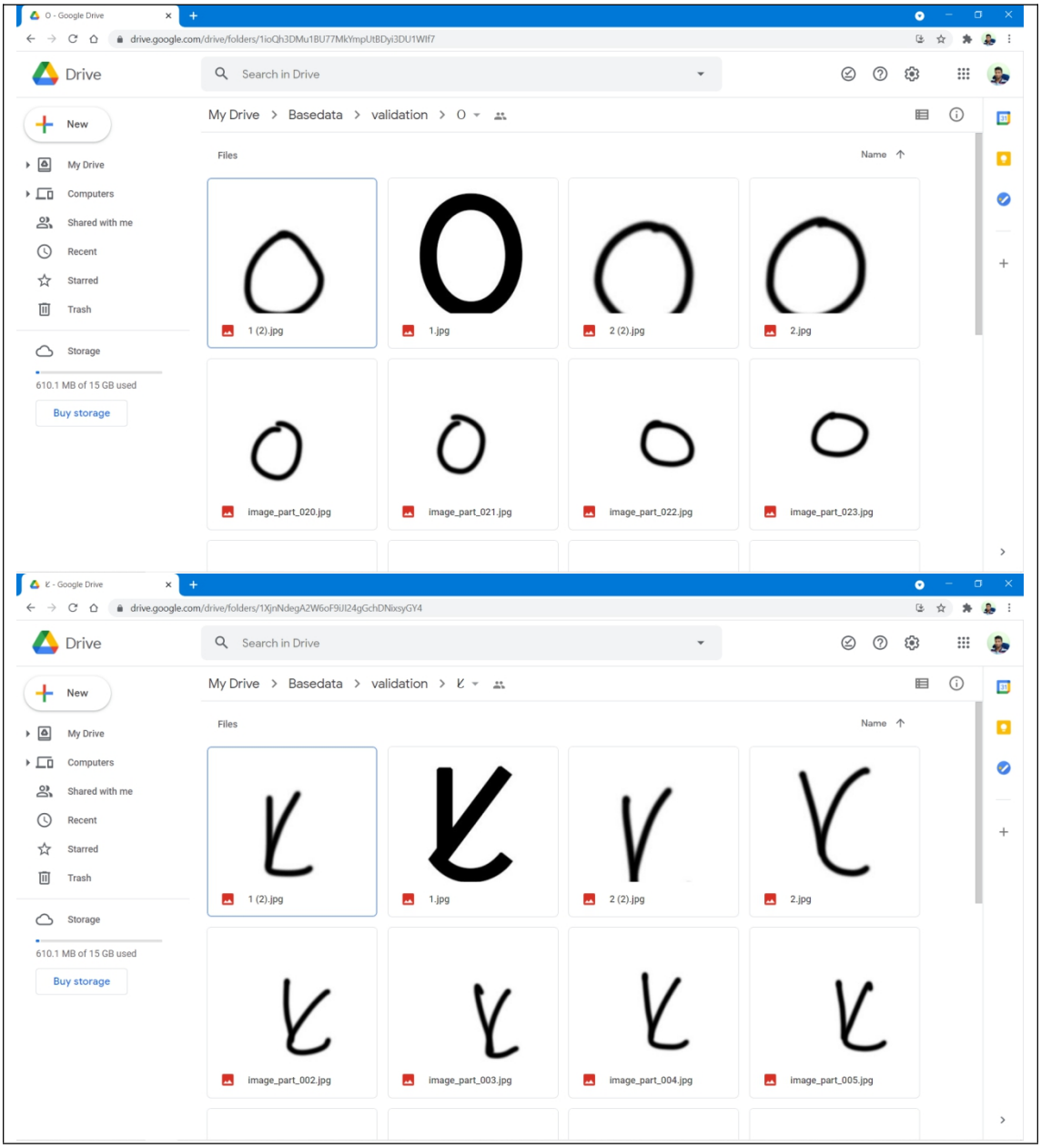
Design methodology

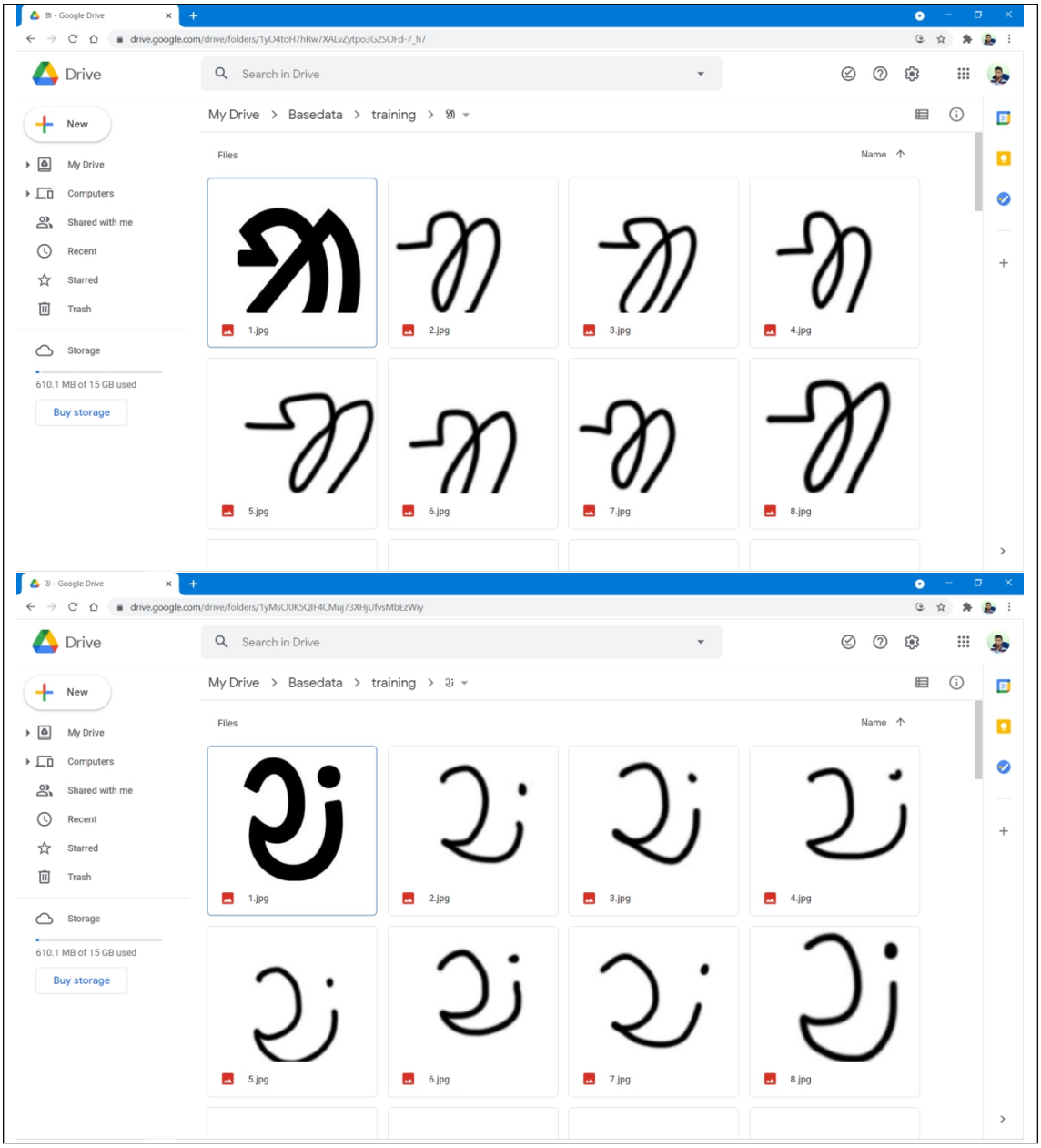
1. The .tflite model and the android application takes image as an input.
2. The trained model acts as a processor that recognizes and extracts the text written in OL Chiki script from the image.
3. It gives output in text form that can be edited i.e. cut, copy, paste and search.

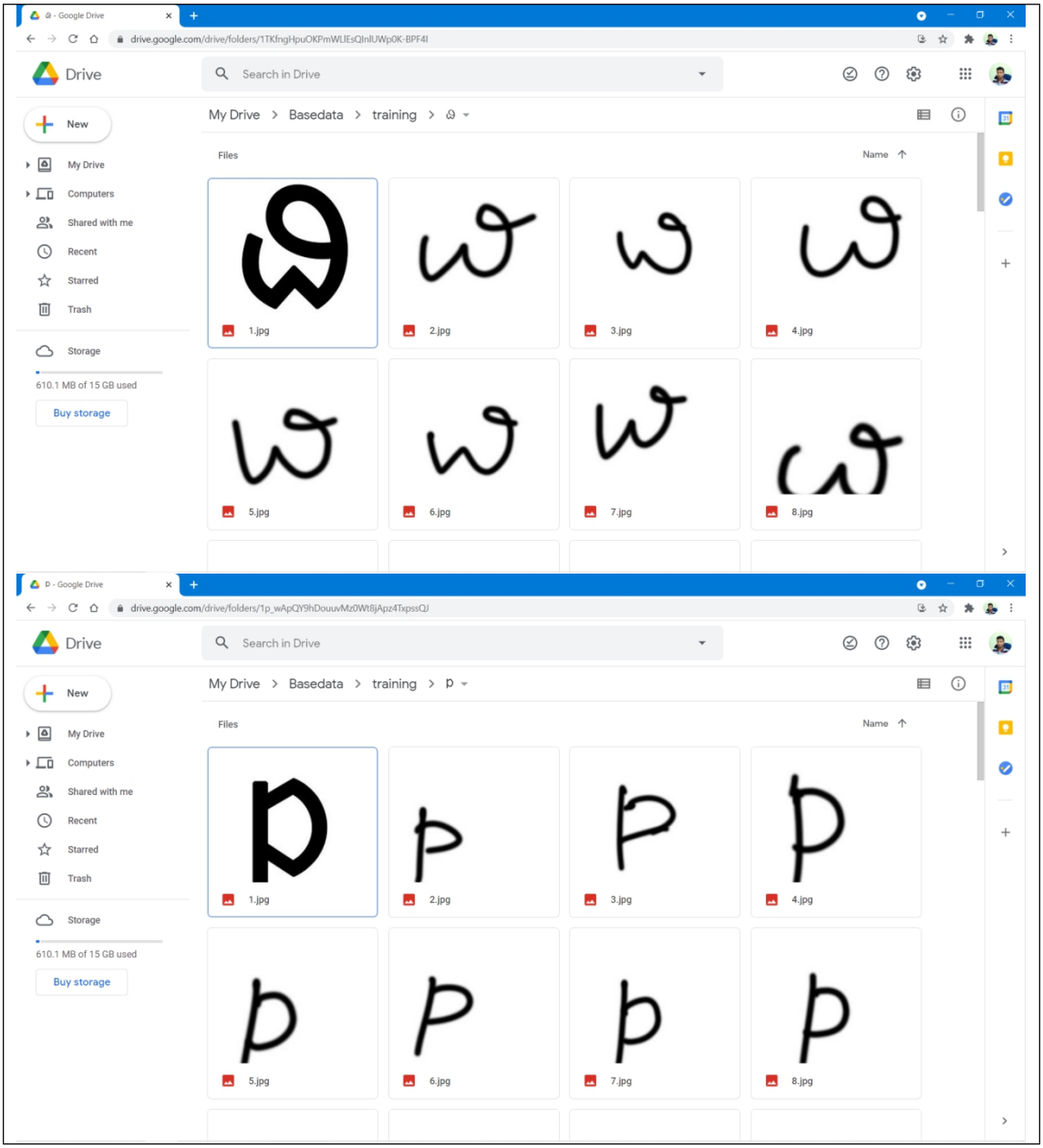


Google Drive

There are 80 images for each letter for training.
There are 20 images for each letter for validation.
There are 2400 images for training.
There are 600 images for validation.




Drive
Search in Drive
New
My Drive
Computers
Shared with me
Recent
Starred
Trash
Storage
610.1 MB of 15 GB used
Buy storage
My Drive > Basedata > training
Files
Name
1.jpg
2.jpg
3.jpg
4.jpg
5.jpg
6.jpg
7.jpg
8.jpg
drive.google.com/drive/folders/1yO4toH7hRw7XALvZytpo3G2SOFd-7_h7
drive.google.com/drive/folders/1yMsCl0K5QIF4CMuj73XHjUfvsMbEzWiy


Drive
Search in Drive
New
My Drive
Computers
Shared with me
Recent
Starred
Trash
Storage
610.1 MB of 15 GB used
Buy storage
My Drive > Basedata > training
Files
Name
1.jpg
2.jpg
3.jpg
4.jpg
5.jpg
6.jpg
7.jpg
8.jpg
My Drive > Basedata > training > ɒ

Coding

Google Colab

.Training the dataset of OL Chiki Script

Importing important libraries

```
try:
  %tensorflow_version 2.x
except:
  pass
```

```
from __future__ import absolute_import, division, print_function, unicode_literals



import tensorflow as tf
import tensorflow_hub as hub
import os
import matplotlib.pyplot as plt
from tensorflow.keras.layers import Dense, Flatten, Conv2D
from tensorflow.keras import Model
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam
from tensorflow.keras import layers
import cv2
import numpy as np



print("\u2022 Using TensorFlow Version:", tf.__version__)
print("\u2022 Using TensorFlow Hub Version: ", hub.__version__)
```

```
print('\u2022 GPU Device Found.' if tf.test.is_gpu_available()
 else '\u2022 GPU Device Not Found. Running on CPU')
```

• Using TensorFlow Version: 2.5.0
• Using TensorFlow Hub Version: 0.12.0
• GPU Device Found.

Mounting the Google Drive

```
data_dir = os.path.join(os.path.dirname("/content/drive/MyDrive/Basedata"), 'Based
ata')
train_dir = os.path.join(data_dir, 'training')
validation_dir = os.path.join(data_dir, 'validation')
```

```
import time
import os
from os.path import exists


def count(dir, counter=0):
    "returns number of files in dir and subdirs"
    for pack in os.walk(dir):
        for f in pack[2]:
            counter += 1
    return dir + " : " + str(counter) + " files"



print('Total images for training :', count(train_dir))
print('Total images for validation :', count(validation_dir))
```

Total images for training : /content/drive/MyDrive/Basedata/training : 2400 files
Total images for validation : /content/drive/MyDrive/Basedata/validation : 600 files

Pretrained Module selection

```
BATCH_SIZE = 32


module_selection = ("mobilenet_v2_100_224", 224, 1280)

handle_base, pixels, FV_SIZE = module_selection

MODULE_HANDLE ="https://tfhub.dev/google/imagenet/{}/feature_vector/4".for
mat(handle_base)

IMAGE_SIZE = (pixels, pixels)

print("Using {} with input size {}, and dimension: {}".format(MODULE_HANDLE,
 IMAGE_SIZE, FV_SIZE))
```

```
Using https://tfhub.dev/google/imagenet/mobilenet_v2_100_224/feature_vector/4
with input size (224, 224), and dimension: 1280
```

Setting image shape to 224x224 px

```
IMAGE_SHAPE = (224, 224)
```

Data Generator

```
datagen_kwargs = dict(rescale=1./255, validation_split=.20)
```

```
valid_datagen = tf.keras.preprocessing.image.ImageDataGenerator(**datagen_kwar
gs)

valid_generator = valid_datagen.flow_from_directory(

validation_dir,subset="validation",shuffle=True,target_size=IMAGE_SHAPE)


train_datagen = tf.keras.preprocessing.image.ImageDataGenerator(**datagen_kwarg
s)
```

```
train_generator = train_datagen.flow_from_directory(
train_dir,subset="training",shuffle=True,target_size=IMAGE_SHAPE)
```

Found 120 images belonging to 30 classes.
Found 1920 images belonging to 30 classes.

Fine Tuning and Model Building

```
do_fine_tuning = True

print("Building model with", MODULE_HANDLE)
model = tf.keras.Sequential([
   tf.keras.layers.InputLayer(input_shape=IMAGE_SIZE + (3,)),
   hub.KerasLayer(MODULE_HANDLE, trainable=do_fine_tuning),
   tf.keras.layers.Dropout(rate=0.2),
   tf.keras.layers.Dense(train_generator.num_classes,
   kernel_regularizer=tf.keras.regularizers.l2(0.001))])

model.build((None,)+IMAGE_SIZE+(3,))
model.summary()
```

Building model with
https://tfhub.dev/google/imagenet/mobilenet_v2_100_224/feature_vector/4
Model: "sequential_2"

Layer (type)	Output Shape	Param #
keras_layer_2 (KerasLayer)	(None, 1280)	2257984

dropout_2 (Dropout) (None, 1280) 0

_____

dense_2 (Dense) (None, 30) 38430

==========

Total params: 2,296,414

Trainable params: 2,262,302

Non-trainable params: 34,112

_____

Model Compilation

```
model.compile(
  optimizer=tf.keras.optimizers.SGD(learning_rate=0.005, momentum=0.9),
  loss=tf.keras.losses.CategoricalCrossentropy(from_logits=True, label_smoothing=
0.1),
  metrics=['accuracy'])
```

Model Training

```
steps_per_epoch = train_generator.samples // train_generator.batch_size
validation_steps = valid_generator.samples // valid_generator.batch_size

EPOCHS =   50
history = model.fit(
    train_generator,
    epochs=EPOCHS, steps_per_epoch=20,
    validation_data=valid_generator,
    validation_steps=validation_steps)
```

Epoch 1/50

20/20 [==============================] - 10s 262ms/step - loss: 2.6054 - accuracy: 0.4187 - val_loss: 3.3427 - val_accuracy: 0.2083

Epoch 2/50

20/20 [==============================] - 5s 226ms/step - loss: 1.0632 - accuracy: 0.9563 - val_loss: 2.8880 - val_accuracy: 0.3021

Epoch 3/50

20/20 [==============================] - 4s 215ms/step - loss: 0.9623 - accuracy: 0.9828 - val_loss: 2.4561 - val_accuracy: 0.4375

Epoch 4/50

20/20 [==============================] - 4s 215ms/step - loss: 0.8917 - accuracy: 0.9969 - val_loss: 2.2244 - val_accuracy: 0.4688

Epoch 5/50

20/20 [==============================] - 4s 220ms/step - loss: 0.8755 - accuracy: 0.9969 - val_loss: 2.1496 - val_accuracy: 0.5312

Epoch 6/50

20/20 [==============================] - 5s 225ms/step - loss: 0.8613 - accuracy: 0.9984 - val_loss: 2.0185 - val_accuracy: 0.5833

Epoch 7/50

20/20 [==============================] - 4s 215ms/step - loss: 0.8643 - accuracy: 0.9937 - val_loss: 1.9610 - val_accuracy: 0.6042

Epoch 8/50

20/20 [==============================] - 5s 227ms/step - loss: 0.8381 - accuracy: 1.0000 - val_loss: 1.7785 - val_accuracy: 0.6458

Epoch 9/50

20/20 [==============================] - 4s 215ms/step - loss: 0.8371 - accuracy: 1.0000 - val_loss: 1.6996 - val_accuracy: 0.7188

Epoch 10/50

20/20 [==============================] - 5s 230ms/step - loss: 0.8301 - accuracy: 1.0000 - val_loss: 1.6548 - val_accuracy: 0.7708

Epoch 11/50

20/20 [==============================] - 5s 228ms/step - loss: 0.8279 - accuracy: 1.0000 - val_loss: 1.5501 - val_accuracy: 0.7812

Epoch 12/50

20/20 [==============================] - 5s 228ms/step - loss: 0.8247 - accuracy: 1.0000 - val_loss: 1.6170 - val_accuracy: 0.7708

Epoch 13/50

20/20 [==============================] - 5s 230ms/step - loss: 0.8247 - accuracy: 1.0000 - val_loss: 1.4834 - val_accuracy: 0.8229

Epoch 14/50

20/20 [==============================] - 5s 226ms/step - loss: 0.8209 - accuracy: 1.0000 - val_loss: 1.4417 - val_accuracy: 0.8542

Epoch 15/50

20/20 [==============================] - 4s 219ms/step - loss: 0.8216 - accuracy: 1.0000 - val_loss: 1.4836 - val_accuracy: 0.8542

Epoch 16/50

20/20 [==============================] - 5s 227ms/step - loss: 0.8156 - accuracy: 1.0000 - val_loss: 1.3720 - val_accuracy: 0.8542

Epoch 17/50

20/20 [==============================] - 5s 228ms/step - loss: 0.8150 - accuracy: 1.0000 - val_loss: 1.3621 - val_accuracy: 0.8750

Epoch 18/50

20/20 [==============================] - 5s 229ms/step - loss: 0.8131 - accuracy: 1.0000 - val_loss: 1.4485 - val_accuracy: 0.8438

Epoch 19/50

20/20 [==============================] - 4s 218ms/step - loss: 0.8134 - accuracy: 1.0000 - val_loss: 1.3956 - val_accuracy: 0.8750

Epoch 20/50

20/20 [==============================] - 4s 221ms/step - loss: 0.8178 - accuracy: 1.0000 - val_loss: 1.2780 - val_accuracy: 0.8958

Epoch 21/50

20/20 [==============================] - 5s 231ms/step - loss: 0.8119 - accuracy: 1.0000 - val_loss: 1.3458 - val_accuracy: 0.9062

Epoch 22/50

20/20 [==============================] - 5s 231ms/step - loss: 0.8104 - accuracy: 1.0000 - val_loss: 1.3159 - val_accuracy: 0.8750

Epoch 23/50

20/20 [==============================] - 5s 229ms/step - loss: 0.8129 - accuracy: 1.0000 - val_loss: 1.5447 - val_accuracy: 0.8125

Epoch 24/50

20/20 [==============================] - 5s 224ms/step - loss: 0.8073 - accuracy: 1.0000 - val_loss: 1.4096 - val_accuracy: 0.8646

Epoch 25/50

20/20 [==============================] - 5s 230ms/step - loss: 0.8062 - accuracy: 1.0000 - val_loss: 1.4072 - val_accuracy: 0.8750

Epoch 26/50

20/20 [==============================] - 5s 232ms/step - loss: 0.8052 - accuracy: 1.0000 - val_loss: 1.4055 - val_accuracy: 0.8646

Epoch 27/50

20/20 [==============================] - 5s 230ms/step - loss: 0.8044 - accuracy: 1.0000 - val_loss: 1.3388 - val_accuracy: 0.8958

Epoch 28/50

20/20 [==============================] - 5s 228ms/step - loss: 0.8063 - accuracy: 1.0000 - val_loss: 1.3437 - val_accuracy: 0.9062

Epoch 29/50

20/20 [==============================] - 5s 231ms/step - loss: 0.8043 - accuracy: 1.0000 - val_loss: 1.3564 - val_accuracy: 0.9271

Epoch 30/50

20/20 [==============================] - 5s 229ms/step - loss: 0.8013 - accuracy: 1.0000 - val_loss: 1.3910 - val_accuracy: 0.8854

Epoch 31/50

20/20 [==============================] - 5s 229ms/step - loss: 0.8014 - accuracy: 1.0000 - val_loss: 1.3219 - val_accuracy: 0.8854

Epoch 32/50

20/20 [==============================] - 4s 221ms/step - loss: 0.8003 - accuracy: 1.0000 - val_loss: 1.2791 - val_accuracy: 0.8958

Epoch 33/50

20/20 [==============================] - 5s 231ms/step - loss: 0.8008 - accuracy: 1.0000 - val_loss: 1.4191 - val_accuracy: 0.8646

Epoch 34/50

20/20 [==============================] - 5s 230ms/step - loss: 0.7994 - accuracy: 1.0000 - val_loss: 1.3840 - val_accuracy: 0.8854

Epoch 35/50

20/20 [==============================] - 5s 231ms/step - loss: 0.7986 - accuracy: 1.0000 - val_loss: 1.3840 - val_accuracy: 0.8854

Epoch 36/50

20/20 [==============================] - 4s 220ms/step - loss: 0.8001 - accuracy: 1.0000 - val_loss: 1.4489 - val_accuracy: 0.8646

Epoch 37/50

20/20 [==============================] - 4s 220ms/step - loss: 0.7991 - accuracy: 1.0000 - val_loss: 1.5256 - val_accuracy: 0.7917

Epoch 38/50

20/20 [==============================] - 5s 226ms/step - loss: 0.7969 - accuracy: 1.0000 - val_loss: 1.5039 - val_accuracy: 0.8229

Epoch 39/50

20/20 [==============================] - 5s 231ms/step - loss: 0.7991 - accuracy: 1.0000 - val_loss: 1.3358 - val_accuracy: 0.9167

Epoch 40/50

20/20 [==============================] - 5s 231ms/step - loss: 0.7962 - accuracy: 1.0000 - val_loss: 1.3014 - val_accuracy: 0.9167

Epoch 41/50

20/20 [==============================] - 5s 225ms/step - loss: 0.7965 - accuracy: 1.0000 - val_loss: 1.3102 - val_accuracy: 0.9167

Epoch 42/50

20/20 [==============================] - 5s 230ms/step - loss: 0.7967 - accuracy: 1.0000 - val_loss: 1.2867 - val_accuracy: 0.9271

Epoch 43/50

20/20 [==============================] - 5s 231ms/step - loss: 0.7962 - accuracy: 1.0000 - val_loss: 1.3337 - val_accuracy: 0.8958

Epoch 45/50

20/20 [==============================] - 5s 230ms/step - loss: 0.7955 - accuracy: 1.0000 - val_loss: 1.3502 - val_accuracy: 0.8958

Epoch 46/50

20/20 [==============================] - 5s 234ms/step - loss: 0.7947 - accuracy: 1.0000 - val_loss: 1.3104 - val_accuracy: 0.8854

Epoch 47/50

20/20 [==============================] - 5s 226ms/step - loss: 0.7937 - accuracy: 1.0000 - val_loss: 1.2922 - val_accuracy: 0.9062

Epoch 48/50

20/20 [==============================] - 5s 232ms/step - loss: 0.7938 - accuracy: 1.0000 - val_loss: 1.2859 - val_accuracy: 0.8958

Epoch 49/50

20/20 [==============================] - 5s 232ms/step - loss: 0.7930 - accuracy: 1.0000 - val_loss: 1.2968 - val_accuracy: 0.8854

Epoch 50/50

20/20 [==============================] - 5s 233ms/step - loss: 0.7942 - accuracy: 1.0000 - val_loss: 1.2425 - val_accuracy: 0.9167

Final Accuracy

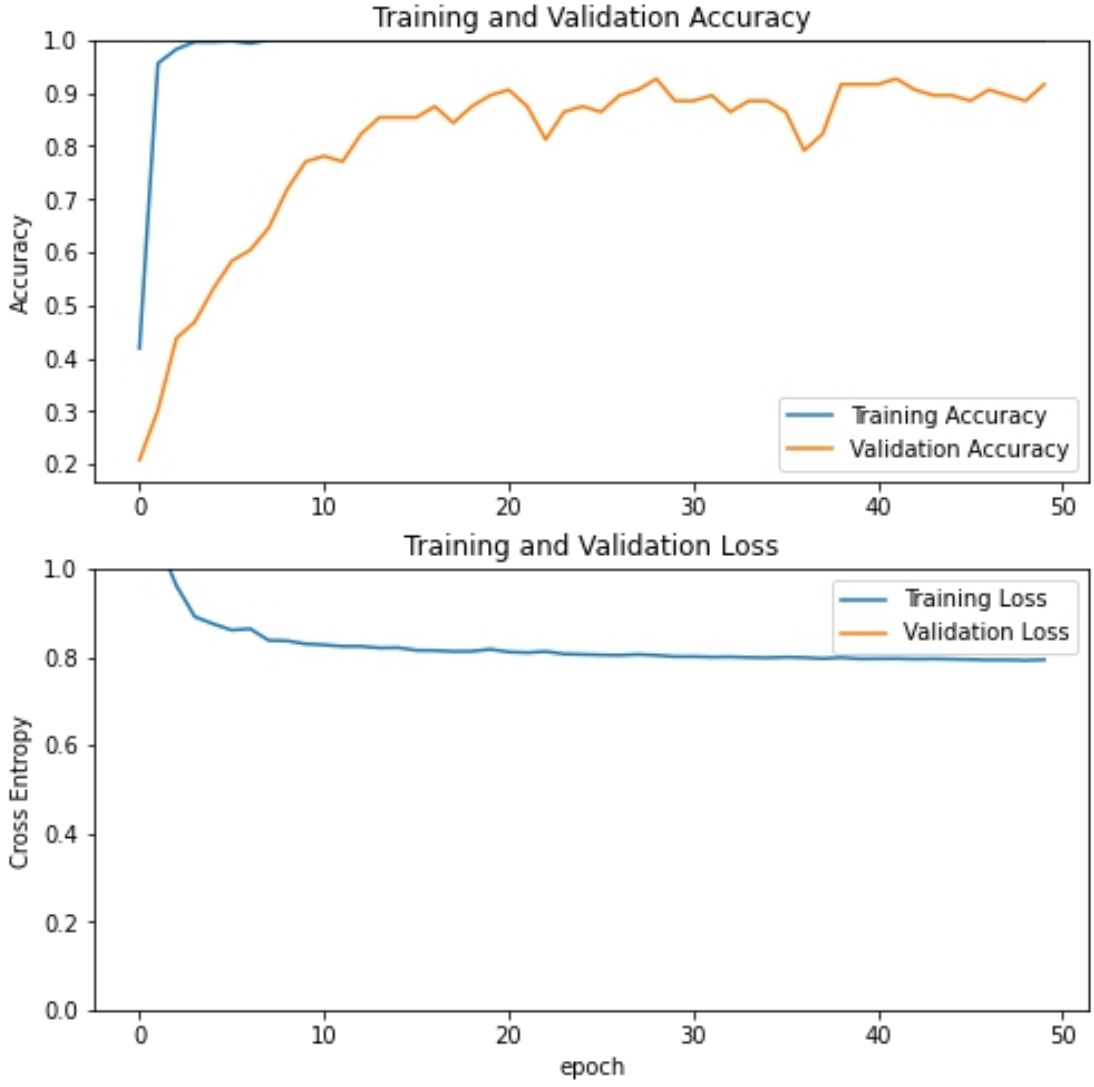
```
final_loss, final_accuracy = model.evaluate(valid_generator, steps = validation_steps)

print()

print("Final accuracy: {:.2f}%".format(final_accuracy * 100))
```

3/3 [==============================] - 0s 90ms/step - loss: 1.2343 - accuracy: 0.9167

Final accuracy: 91.67%

Training Graph

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

loss = history.history['loss']
val_loss = history.history['val_loss']

plt.figure(figsize=(8, 8))
plt.subplot(2, 1, 1)
plt.plot(acc, label='Training Accuracy')
plt.plot(val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
plt.ylabel('Accuracy')
plt.ylim([min(plt.ylim()),1])
plt.title('Training and Validation Accuracy')
 plt.subplot(2, 1, 2)
plt.plot(loss, label='Training Loss')
plt.plot(val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.ylabel('Cross Entropy')
plt.ylim([0,1.0])
plt.title('Training and Validation Loss')
plt.xlabel('epoch')
plt.show()
```


Training and Validation Accuracy
Accuracy
Training Accuracy
Validation Accuracy
Training and Validation Loss
Cross Entropy
Training Loss
Validation Loss
epoch

Label Generator

```
classes = []
with open("/content/drive/MyDrive/Basedata/labels.txt") as file:
    for l in file:
        classes.append(l.strip())


print(classes)
print("Total Characters ",len(classes))
```

```
['ಀ ', 'ಁ ', 'ಂ ', 'ಃ ', '಄ ', 'ಅ ', 'ಆ ', 'ಇ ', 'ಈ ', 'ಉ ', 'ಊ ', 'ಋ ', 'ಌ ', '಍ ', 'ಎ ', 'ಏ ', 'ಐ ', '಑ ', 'ಒ ', 'ಓ ', 'ಔ ', 'ಕ ', 'ಖ ', 'ಗ ', 'ಘ ', 'ಙ ']
Total Characters  30
```

```
print(train_generator.class_indices)
```

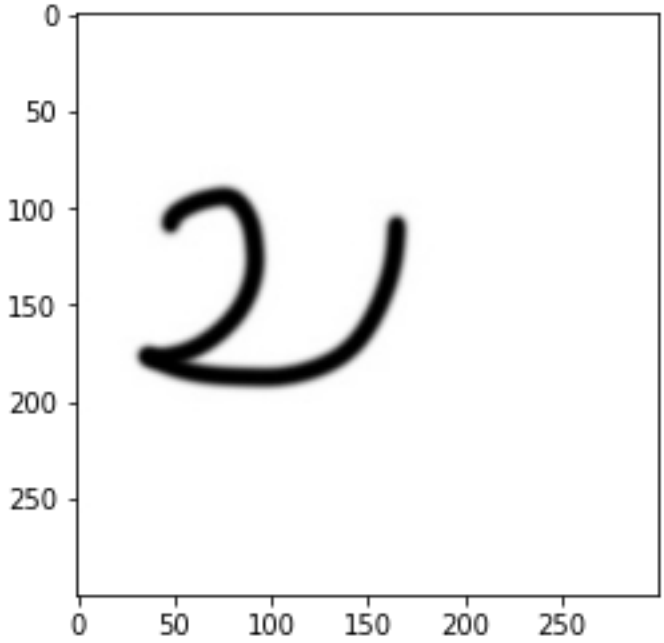
```
{'ಀ ': 0, 'ಁ ': 1, 'ಂ ': 2, 'ಃ ': 3, '಄ ': 4, 'ಅ ': 5, 'ಆ ': 6, 'ಇ ': 7, 'ಈ ': 8, 'ಉ ': 9, 'ಊ ': 10, 'ಋ ': 11, 'ಌ ': 12, '಍ ': 13, 'ಎ ': 14, 'ಏ ': 15, 'ಐ ': 16, '಑ ': 17, 'ಒ ': 18, 'ಓ ': 19, 'ಔ ': 20, 'ಕ ': 21, 'ಖ ': 22, 'ಗ ': 23, 'ಘ ': 24, 'ಙ ': 25, 'ಚ ': 26, 'ಛ ': 27, 'ಜ ': 28, 'ಝ ': 29}
```

Prediction

```
img_file = '/content/drive/MyDrive/Basedata/training/ಀ /image_part_018.jpg'


img = cv2.imread(img_file)
plt.imshow(img)
img = cv2.resize(img, (224, 224) )
img = img /255
```

```
img_class = model.predict(np.asarray([img]))[0]
class_idx = np.argmax(img_class)

print("Character is : %s"%(classes[class_idx]))
print(f"Index: {class_idx}")
print("Accuracy: %f"%(img_class[class_idx] * 10))
```

Character is : ฮ
Index: 9
Accuracy: 58.475924



Conversion to ".tflite" model

```
converter=tf.lite.TFLiteConverter.from_keras_model(model)
tflite_float_model=converter.convert()

float_model_size=len(tflite_float_model) / (1024 * 1024)
print()
print('.tflite model size = %d MB' % float_model_size)
f=open('model.tflite', "wb")
```

```
f.write(tflite_float_model)
f.close()
```

```
INFO:tensorflow:Assets written to: /tmp/tmpydr17qjj/assets
INFO:tensorflow:Assets written to: /tmp/tmpydr17qjj/assets
```

.tflite model size = 8 MB

Optimization of the model

```
converter.optimizations=[tf.lite.Optimize.DEFAULT]
tflite_optimized_model=converter.convert()


optimized_model_size=len(tflite_optimized_model) / (1024 * 1024)
print()
print('Optimized .tflite model size = %d MB'%optimized_model_size)
print('It is about %d%% of the .tflite model size.'\
    % (optimized_model_size * 100 / float_model_size))
```

```
INFO:tensorflow:Assets written to: /tmp/tmppm8u4j8a/assets
INFO:tensorflow:Assets written to: /tmp/tmppm8u4j8a/assets
```

Optimized .tflite model size = 2 MB
It is about 30% of the .tflite model size.

Downloads

```
f=open('model.tflite', "wb")
f.write(tflite_optimized_model)
f.close()
from google.colab import files
files.download('model.tflite')
```

Android Studio

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.olchikiocr">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Ol Chiki Ocr"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.EverGreenCrop">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Classifier.kt

```
package com.example.olchikiocr

import android.content.res.AssetManager
import android.graphics.Bitmap
import android.util.Log
import org.tensorflow.lite.Interpreter
import java.io.FileInputStream
import java.nio.ByteBuffer
import java.nio.ByteOrder
import java.nio.MappedByteBuffer
import java.nio.channels.FileChannel
import java.util.*
import kotlin.math.min

class Classifier(assetManager: AssetManager, modelPath: String, labelPath: String,
private val inputSize: Int) {
    private var interpreter: Interpreter
    private var lableList: List<String>
    private val pixelSize: Int = 3
    private val imageMean = 0
    private val imageStd = 255.0f
```

```
    private val maxResult = 3
    private val threshHold = 0.4f

    data class Recognition(
        var id: String = "",
        var title: String = "",
        var confidence: Float = 0F
    ) {
        override fun toString(): String {
            return "Title = $title, Confidence = $confidence)"
        }
    }

    init {
        val options = Interpreter.Options()
            options.setNumThreads(2)
        options.setUseNNAPI(true)
        interpreter = Interpreter(loadModelFile(assetManager, modelPath), options)
        lableList = loadLabelList(assetManager, labelPath)
    }

    private fun loadModelFile(assetManager: AssetManager, modelPath: String):
MappedByteBuffer {
        val fileDescriptor = assetManager.openFd(modelPath)
        val inputStream = FileInputStream(fileDescriptor.fileDescriptor)
        val fileChannel = inputStream.channel
        val startOffset = fileDescriptor.startOffset
        val declaredLength = fileDescriptor.declaredLength
        return fileChannel.map(FileChannel.MapMode.READ_ONLY, startOffset,
declaredLength)
    }

    private fun loadLabelList(assetManager: AssetManager, labelPath: String):
List<String> {
        return assetManager.open(labelPath).bufferedReader().useLines { it.toList() }

    }

        fun recognizeImage(bitmap: Bitmap): List<Recognition> {
        val scaledBitmap = Bitmap.createScaledBitmap(bitmap, inputSize, inputSize,
false)
        val byteBuffer = convertBitmapToByteBuffer(scaledBitmap)
        val result = Array(1) { FloatArray(lableList.size) }
        interpreter.run(byteBuffer, result)
        return getSortedResult(result)
    }


    private fun convertBitmapToByteBuffer(bitmap: Bitmap): ByteBuffer {
        val byteBuffer = ByteBuffer.allocateDirect(4 * inputSize * inputSize *
```

```
pixelSize)
        byteBuffer.order(ByteOrder.nativeOrder())
        val intValues = IntArray(inputSize * inputSize)

        bitmap.getPixels(intValues, 0, bitmap.width, 0, 0, bitmap.width, bitmap.height)
        var pixel = 0
        for (i in 0 until inputSize) {
            for (j in 0 until inputSize) {
                val input = intValues[pixel++]

                byteBuffer.putFloat((((input.shr(16)  and 0xFF) - imageMean) / imageStd))
                byteBuffer.putFloat((((input.shr(8) and 0xFF) - imageMean) / imageStd))
                byteBuffer.putFloat((((input and 0xFF) - imageMean) / imageStd))
            }
        }
        return byteBuffer
    }

    private fun getSortedResult(labelProbArray: Array<FloatArray>):
List<Recognition> {
        Log.d("Classifier", "List
Size:(%d, %d, %d)".format(labelProbArray.size,labelProbArray[0].size,lableList.si
ze))

        val pq = PriorityQueue(
            maxResult,
            Comparator<Recognition> {
                    (_, _, confidence1), (_, _, confidence2)
                -> confidence1.compareTo(confidence2) * -1
            })

        for (i in lableList.indices) {
            val confidence = labelProbArray[0][i]
            if (confidence >= threshHold) {
                pq.add(Recognition("" + i,
                    if (lableList.size > i) lableList[i] else "Unknown", confidence*10)
                )
            }
        }
        Log.d("Classifier", "pqsize:(%d)".format(pq.size))

        val recognitions = ArrayList<Recognition>()
        val recognitionsSize = min(pq.size, maxResult)
        for (i in 0 until recognitionsSize) {
            recognitions.add(pq.poll())
        }
        return recognitions
    }

}
```

MainActivity.kt

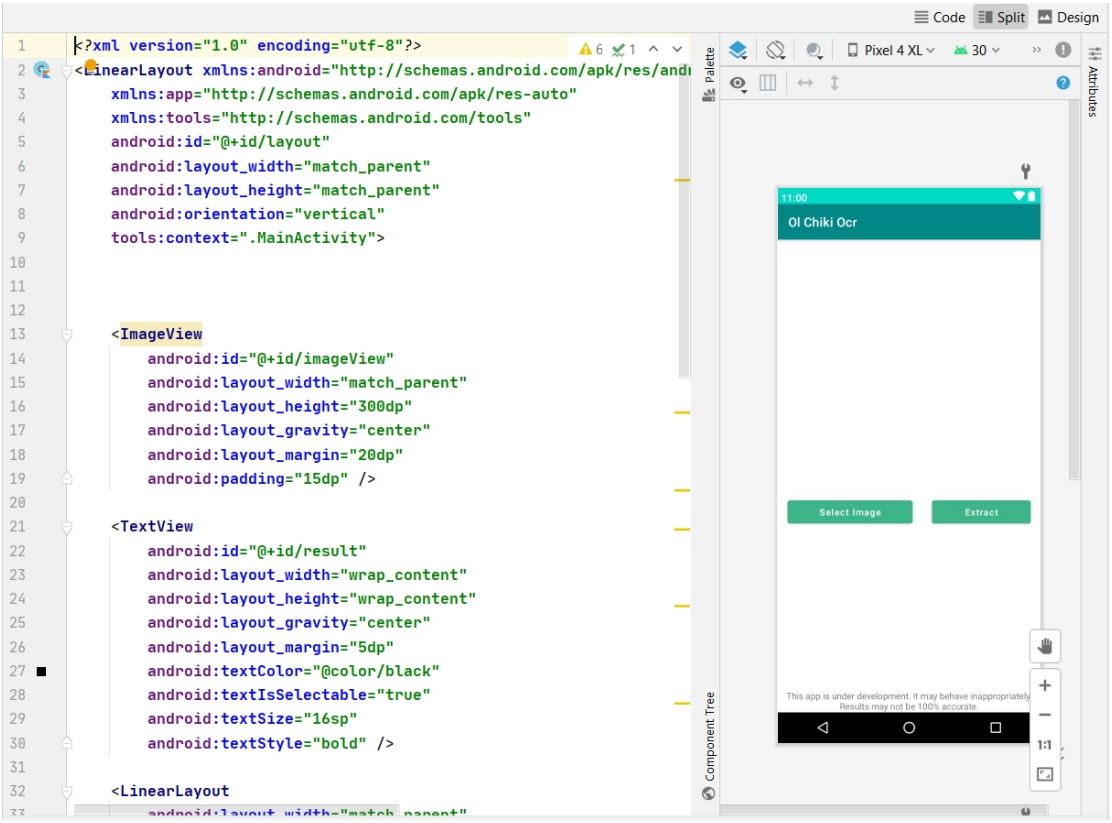
```
package com.example.olchikiocr

import android.annotation.SuppressLint
import android.content.Intent
import android.graphics.Bitmap
import android.graphics.drawable.BitmapDrawable
import android.net.Uri
import android.os.Bundle
import android.provider.MediaStore
import android.view.View
import android.widget.Button
import android.widget.ImageView
import android.widget.LinearLayout
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import androidx.appcompat.app.AppCompatDelegate
import com.example.olchikiocr.databinding.ActivityMainBinding

class MainActivity : AppCompatActivity(), View.OnClickListener {
    lateinit var bitmap: Bitmap

    //final version for university
    private val mGalleryRequestCode = 2
    private val mInputSize = 224

    private val mModelPath = "project.tflite"
    private val mLabelPath = "labels.txt"

    private lateinit var classifier: Classifier
    private lateinit var binding: ActivityMainBinding

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        title = "OL CHIKI OCR"
        val root: LinearLayout = findViewById(R.id.layout)
        binding = ActivityMainBinding.bind(root)


AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_F
OLLOW_SYSTEM)
        initClassifier()
        initViews()


        binding.gallery.setOnClickListener {
            val callGalleryIntent = Intent(Intent.ACTION_PICK)
```

```
            callGalleryIntent.type = "image/*"
            startActivityForResult(callGalleryIntent, mGalleryRequestCode)

        }
    }

    override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
        super.onActivityResult(requestCode, resultCode, data)

        if(requestCode == mGalleryRequestCode) {
            binding.imageView.setImageURI(data?.data)
            val ur: Uri? = data?.data

            bitmap = MediaStore.Images.Media.getBitmap(this.contentResolver, ur)

        } else {
            Toast.makeText(this, "Unrecognized request code",
Toast.LENGTH_LONG).show()

        }
    }

    private fun initClassifier() {
        classifier = Classifier(assets, mModelPath, mLabelPath, mInputSize)
    }

    private fun initViews() {

            findViewById<ImageView>(R.id.imageView).setOnClickListener(this)

    }

    @SuppressLint("SetTextI18n")
    override fun onClick(view: View?) {
        bitmap = ((view as ImageView).drawable as BitmapDrawable).bitmap

        val result = classifier.recognizeImage(bitmap)

        runOnUiThread {

            Toast.makeText(this, result[0].title+"\n"+result[0].confidence,
Toast.LENGTH_SHORT).show()
            binding.result.text = result[0].title+"\n"+result[0].confidence
        }
    }
}
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">



    <ImageView
        android:id="@+id/imageView"
        android:layout_width="match_parent"
        android:layout_height="300dp"
        android:layout_gravity="center"
        android:layout_margin="20dp"
        android:padding="15dp" />

    <TextView
        android:id="@+id/result"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_margin="5dp"
        android:textColor="@color/black"
        android:textIsSelectable="true"
        android:textSize="16sp"
        android:textStyle="bold" />

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <Button
            android:id="@+id/gallery"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="15dp"
            android:layout_marginTop="30dp"
            android:layout_marginRight="15dp"
            android:layout_weight="1"
            android:text="Select Image"
            android:textAllCaps="false"
            app:backgroundTint="#3EB489" />
```

```
    <Button
        android:id="@+id/extr"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="15dp"
        android:layout_marginTop="30dp"
        android:layout_marginRight="15dp"
        android:layout_weight="1"
        android:text="Extract"
        android:textAllCaps="false"
        app:backgroundTint="#3EB489" />

  </LinearLayout>

  <TextView
      android:layout_width="wrap_content"
      android:layout_height="match_parent"
      android:gravity="bottom"
      android:text="This app is under development. It may behave
inappropriately. Results may not be 100% accurate."
      android:textAlignment="center"
      tools:ignore="RtlCompat" />


</LinearLayout>
```



ExampleInstrumentedTest

```
package com.example.olchikiocr

import androidx.test.platform.app.InstrumentationRegistry
import androidx.test.ext.junit.runners.AndroidJUnit4

import org.junit.Test
import org.junit.runner.RunWith

import org.junit.Assert.*

/**
 * Instrumented test, which will execute on an Android device.
 *
 * See [testing documentation](http://d.android.com/tools/testing).
 */
@RunWith(AndroidJUnit4::class)
class ExampleInstrumentedTest {
    @Test
    fun useAppContext() {
        // Context of the app under test.
        val appContext = InstrumentationRegistry.getInstrumentation().targetContext
        assertEquals("com.example.olchikiocr", appContext.packageName)
    }
}
```

ExampleUnitTest

```
package com.example.olchikiocr

import org.junit.Test

import org.junit.Assert.*

/**
 * Example local unit test, which will execute on the development machine (host).
 *
 * See [testing documentation](http://d.android.com/tools/testing).
 */
class ExampleUnitTest {
    @Test
    fun addition_isCorrect() {
        assertEquals(4, 2 + 2)
    }
}
```

BuildConfig

```
/**
 * Automatically generated file. DO NOT MODIFY
 */
package com.example.olchikiocr;

public final class BuildConfig {
  public static final boolean DEBUG = Boolean.parseBoolean("true");
  public static final String APPLICATION_ID = "com.example.olchikiocr";
  public static final String BUILD_TYPE = "debug";
  public static final int VERSION_CODE = 1;
  public static final String VERSION_NAME = "1.0";
}
```

ic_launcher_background.xml

```
<?xml version="1.0" encoding="utf-8"?>
<vector xmlns:android="http://schemas.android.com/apk/res/android"
   android:width="108dp"
   android:height="108dp"
   android:viewportWidth="108"
   android:viewportHeight="108">
  <group android:scaleX="0"
     android:scaleY="0"
     android:translateX="54"
     android:translateY="54">
     <path android:fillColor="#3DDC84"
          android:pathData="M0,0h108v108h-108z"/>
     <path android:fillColor="#00000000" android:pathData="M9,0L9,108"
          android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
     <path android:fillColor="#00000000" android:pathData="M19,0L19,108"
          android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
     <path android:fillColor="#00000000" android:pathData="M29,0L29,108"
          android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
     <path android:fillColor="#00000000" android:pathData="M39,0L39,108"
          android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
     <path android:fillColor="#00000000" android:pathData="M49,0L49,108"
          android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
     <path android:fillColor="#00000000" android:pathData="M59,0L59,108"
          android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
     <path android:fillColor="#00000000" android:pathData="M69,0L69,108"
          android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
     <path android:fillColor="#00000000" android:pathData="M79,0L79,108"
          android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
     <path android:fillColor="#00000000" android:pathData="M89,0L89,108"
```

```
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M99,0L99,108"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M0,9L108,9"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M0,19L108,19"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M0,29L108,29"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M0,39L108,39"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M0,49L108,49"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M0,59L108,59"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M0,69L108,69"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M0,79L108,79"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M0,89L108,89"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M0,99L108,99"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M19,29L89,29"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M19,39L89,39"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M19,49L89,49"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M19,59L89,59"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M19,69L89,69"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M19,79L89,79"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M29,19L29,89"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M39,19L39,89"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M49,19L49,89"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M59,19L59,89"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M69,19L69,89"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
    <path android:fillColor="#00000000" android:pathData="M79,19L79,89"
        android:strokeColor="#33FFFFFF" android:strokeWidth="0.8"/>
  </group>
</vector>
```

ic_launcher_foreground.xml

```
<vector xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:aapt="http://schemas.android.com/aapt"
    android:width="108dp"
    android:height="108dp"
    android:viewportWidth="108"
    android:viewportHeight="108">
    <path android:pathData="M31,63.928c0,0 6.4,-11 12.1,-13.1c7.2,-2.6 26,-1.4
26,-1.4l38.1,38.1L107,108.928l-32,-1L31,63.928z">
        <aapt:attr name="android:fillColor">
            <gradient
                android:endX="85.84757"
                android:endY="92.4963"
                android:startX="42.9492"
                android:startY="49.59793"
                android:type="linear">
                <item
                    android:color="#44000000"
                    android:offset="0.0" />
                <item
                    android:color="#00000000"
                    android:offset="1.0" />
            </gradient>
        </aapt:attr>
    </path>
    <path
        android:fillColor="#FFFFFF"
        android:fillType="nonZero"
        android:pathData="M65.3,45.828l3.8,-6.6c0.2,-0.4 0.1,-0.9 -0.3,-1.1c-0.4,-0.2
-0.9,-0.1 -1.1,0.3l-3.9,6.7c-6.3,-2.8 -13.4,-2.8 -19.7,0l-3.9,-6.7c-0.2,-0.4 -0.7,-0.5 -
1.1,-0.3C38.8,38.328 38.7,38.828 38.9,39.228l3.8,6.6C36.2,49.428 31.7,56.028
31,63.928h46C76.3,56.028 71.8,49.428 65.3,45.828zM43.4,57.328c-0.8,0 -1.5,-0.5 -
1.8,-1.2c-0.3,-0.7 -0.1,-1.5 0.4,-2.1c0.5,-0.5 1.4,-0.7 2.1,-0.4c0.7,0.3 1.2,1
1.2,1.8C45.3,56.528 44.5,57.328 43.4,57.328L43.4,57.328zM64.6,57.328c-0.8,0 -
1.5,-0.5 -1.8,-1.2s-0.1,-1.5 0.4,-2.1c0.5,-0.5 1.4,-0.7 2.1,-0.4c0.7,0.3 1.2,1
1.2,1.8C66.5,56.528 65.6,57.328 64.6,57.328L64.6,57.328z"
        android:strokeWidth="1"
        android:strokeColor="#00000000" />
</vector>
```

ic_launcher.xml

```
<?xml version="1.0" encoding="utf-8"?>
<adaptive-icon xmlns:android="http://schemas.android.com/apk/res/android">
    <background android:drawable="@color/ic_launcher_background"/>
    <foreground android:drawable="@mipmap/ic_launcher_foreground"/>
</adaptive-icon>
```

ic_launcher_round.xml

```
<?xml version="1.0" encoding="utf-8"?>
<adaptive-icon xmlns:android="http://schemas.android.com/apk/res/android">
    <background android:drawable="@color/ic_launcher_background"/>
    <foreground android:drawable="@mipmap/ic_launcher_foreground"/>
</adaptive-icon>
```

colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="purple_200">#FFBB86FC</color>
    <color name="purple_500">#FF6200EE</color>
    <color name="purple_700">#FF3700B3</color>
    <color name="teal_200">#FF03DAC5</color>
    <color name="teal_700">#FF018786</color>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>
</resources>
```

ic_launcher_background.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="ic_launcher_background">#FFFFFF</color>
</resources>
```

strings.xml

```
<resources>
    <string name="app_name">EverGreenCrop</string>
</resources>
```

themes.xml

```
<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
    <style name="Theme.EverGreenCrop"
parent="Theme.MaterialComponents.DayNight.DarkActionBar">
        <!-- Primary brand color. -->
        <item name="colorPrimary">@color/teal_700</item>
        <item name="colorPrimaryVariant">@color/teal_200</item>
        <item name="colorOnPrimary">@color/white</item>
        <!-- Secondary brand color. -->
        <item name="colorSecondary">@color/teal_200</item>
        <item name="colorSecondaryVariant">@color/teal_700</item>
```

```
        <item name="colorOnSecondary">@color/black</item>
        <!-- Status bar color. -->
        <item name="android:statusBarColor"
tools:targetApi="l">?attr/colorPrimaryVariant</item>
        <!-- Customize your theme here. -->
    </style>
</resources>
```

themes.xml

```
<resources xmlns:tools="http://schemas.android.com/tools">
    <!-- Base application theme. -->
    <style name="Theme.EverGreenCrop"
parent="Theme.MaterialComponents.DayNight.DarkActionBar">
        <!-- Primary brand color. -->
        <item name="colorPrimary">@color/teal_700</item>
        <item name="colorPrimaryVariant">@color/teal_200</item>
        <item name="colorOnPrimary">@color/white</item>
        <!-- Secondary brand color. -->
        <item name="colorSecondary">@color/teal_200</item>
        <item name="colorSecondaryVariant">@color/teal_200</item>
        <item name="colorOnSecondary">@color/white</item>
        <!-- Status bar color. -->
        <item name="android:statusBarColor"
tools:targetApi="l">?attr/colorPrimaryVariant</item>
        <!-- Customize your theme here. -->
    </style>
</resources>
```

build.gradle

```
// Top-level build file where you can add configuration options common to all sub-
projects/modules.
buildscript {
    ext.kotlin_version = "1.4.31"
    repositories {
        google()
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:4.2.1'
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"

        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
    }
}
```

```
allprojects {
    repositories {
        google()
        jcenter()
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

build.gradle

```
plugins {
    id 'com.android.application'
    id 'kotlin-android'
}

android {
    compileSdkVersion 30
    buildToolsVersion "30.0.3"

    defaultConfig {
        applicationId "com.example.olchikiocr"
        minSdkVersion 16
        targetSdkVersion 30
        versionCode 1
        versionName "1.0"

        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }

    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'),
'proguard-rules.pro'
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
    kotlinOptions {
        jvmTarget = '1.8'
    }
    aaptOptions {
        noCompress "tflite"
        noCompress "lite"
```

```
    }
    buildFeatures {
        viewBinding true
    }
}

dependencies {

    implementation "org.jetbrains.kotlin:kotlin-stdlib:$kotlin_version"
    implementation 'androidx.core:core-ktx:1.3.2'
    implementation 'androidx.appcompat:appcompat:1.2.0'
    implementation 'com.google.android.material:material:1.3.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
    testImplementation 'junit:junit:4.+'
    androidTestImplementation 'androidx.test.ext:junit:1.1.2'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'

    implementation 'org.tensorflow:tensorflow-lite:2.4.0'
}
```

gradle-wrapper.properties

```
#Mon Apr 19 11:47:13 IST 2021
distributionBase=GRADLE_USER_HOME
distributionPath=wrapper/dists
zipStoreBase=GRADLE_USER_HOME
zipStorePath=wrapper/dists
distributionUrl=https\://services.gradle.org/distributions/gradle-6.7.1-all.zip
```

proguard-rules.pro

```
# Add project specific ProGuard rules here.
# You can control the set of applied configuration files using the
# proguardFiles setting in build.gradle.
#
# For more details, see
#   http://developer.android.com/guide/developing/tools/proguard.html

# If your project uses WebView with JS, uncomment the following
# and specify the fully qualified class name to the JavaScript interface
# class:
#-keepclassmembers class fqcn.of.javascript.interface.for.webview {
#   public *;
#}

# Uncomment this to preserve the line number information for
# debugging stack traces.
#-keepattributes SourceFile,LineNumberTable
```

```
# If you keep the line number information, uncomment this to
# hide the original source file name.
#-renamesourcefileattribute SourceFile
```

gradle.properties

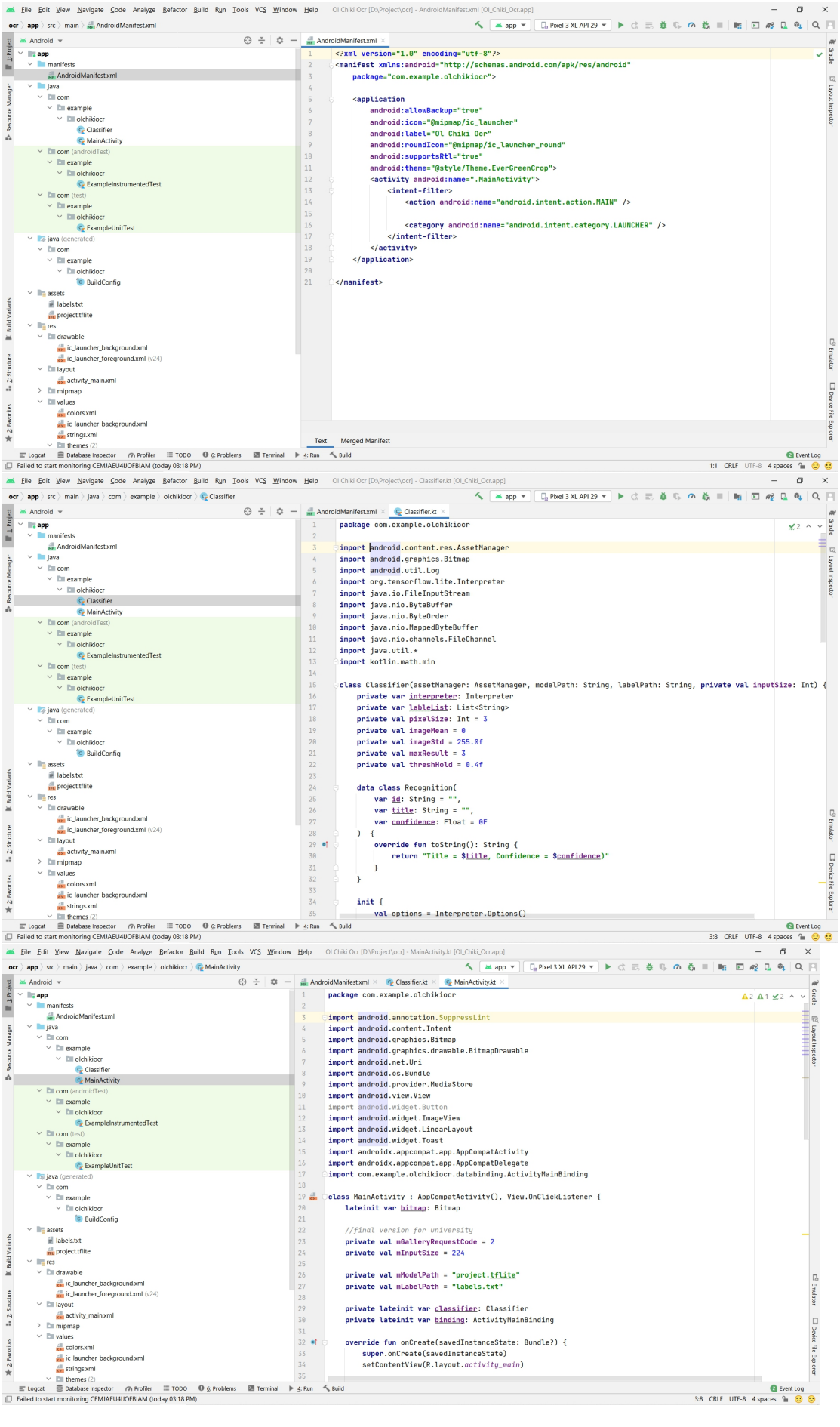
```
# Project-wide Gradle settings.
# IDE (e.g. Android Studio) users:
# Gradle settings configured through the IDE *will override*
# any settings specified in this file.
# For more details on how to configure your build environment visit
# http://www.gradle.org/docs/current/userguide/build_environment.html
# Specifies the JVM arguments used for the daemon process.
# The setting is particularly useful for tweaking memory settings.
org.gradle.jvmargs=-Xmx2048m -Dfile.encoding=UTF-8
# When configured, Gradle will run in incubating parallel mode.
# This option should only be used with decoupled projects. More details, visit
#
http://www.gradle.org/docs/current/userguide/multi_project_builds.html#sec:decoupled_projects
# org.gradle.parallel=true
# AndroidX package structure to make it clearer which packages are bundled with the
# Android operating system, and which are packaged with your app"s APK
# https://developer.android.com/topic/libraries/support-library/androidx-rn
android.useAndroidX=true
# Automatically convert third-party libraries to use AndroidX
android.enableJetifier=true
# Kotlin code style for this project: "official" or "obsolete":
kotlin.code.style=official
```

settings.gradle

```
include ':app'
rootProject.name = "Ol Chiki Ocr"
```

local.properties

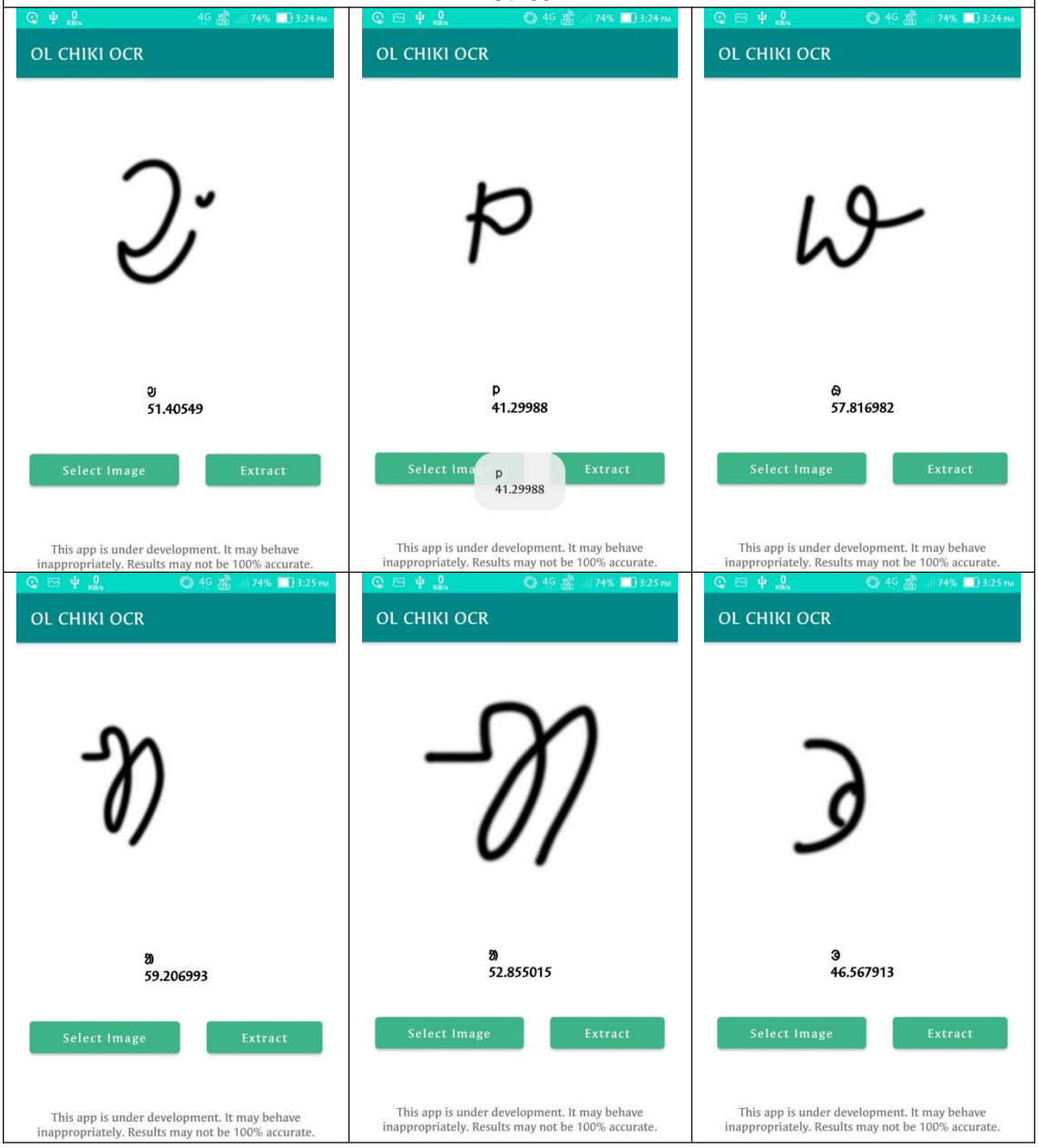
```
## This file must *NOT* be checked into Version Control Systems,
# as it contains information specific to your local configuration.
#
# Location of the SDK. This is only used by Gradle.
# For customization when using a Version Control System, please read the
# header note.
#Mon Apr 19 11:45:53 IST 2021
sdk.dir=C\:\\Users\\smsun\\AppData\\Local\\Android\\Sdk
```

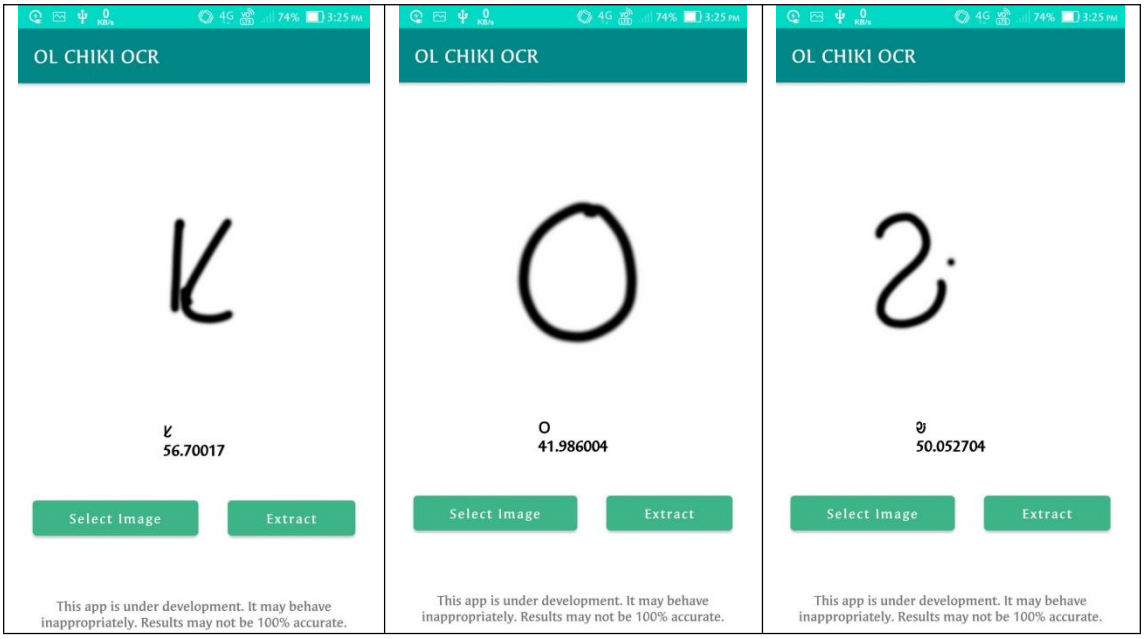

AndroidManifest.xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="com.example.olchikiocr">
<application
android:allowBackup="true"
android:icon="@mipmap/ic_launcher"
android:label="Ol Chiki Ocr"
android:roundIcon="@mipmap/ic_launcher_round"
android:supportsRtl="true"
android:theme="@style/Theme.EverGreenCrop">
<activity android:name=".MainActivity">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
</application>
</manifest>
Classifier.kt
package com.example.olchikiocr
import android.content.res.AssetManager
import android.graphics.Bitmap
import android.util.Log
import org.tensorflow.lite.Interpreter
import java.io.FileInputStream
import java.nio.ByteBuffer
import java.nio.ByteOrder
import java.nio.MappedByteBuffer
import java.nio.channels.FileChannel
import java.util.*
import kotlin.math.min
class Classifier(assetManager: AssetManager, modelPath: String, labelPath: String, private val inputSize: Int) {
private var interpreter: Interpreter
private var lableList: List<String>
private val pixelSize: Int = 3
private val imageMean = 0
private val imageStd = 255.0f
private val maxResult = 3
private val threshHold = 0.4f
data class Recognition(
var id: String = "",
var title: String = "",
var confidence: Float = 0F
) {
override fun toString(): String {
return "Title = $title, Confidence = $confidence)"
}
}
init {
val options = Interpreter.Options()
MainActivity.kt
package com.example.olchikiocr
import android.annotation.SuppressLint
import android.content.Intent
import android.graphics.Bitmap
import android.graphics.drawable.BitmapDrawable
import android.net.Uri
import android.os.Bundle
import android.provider.MediaStore
import android.view.View
import android.widget.Button
import android.widget.ImageView
import android.widget.LinearLayout
import android.widget.Toast
import androidx.appcompat.app.AppCompatActivity
import androidx.appcompat.app.AppCompatDelegate
import com.example.olchikiocr.databinding.ActivityMainBinding
class MainActivity : AppCompatActivity(), View.OnClickListener {
lateinit var bitmap: Bitmap
//final version for university
private val mGalleryRequestCode = 2
private val mInputSize = 224
private val mModelPath = "project.tflite"
private val mLabelPath = "labels.txt"
private lateinit var classifier: Classifier
private lateinit var binding: ActivityMainBinding
override fun onCreate(savedInstanceState: Bundle?) {
super.onCreate(savedInstanceState)
setContentView(R.layout.activity_main)

Testing

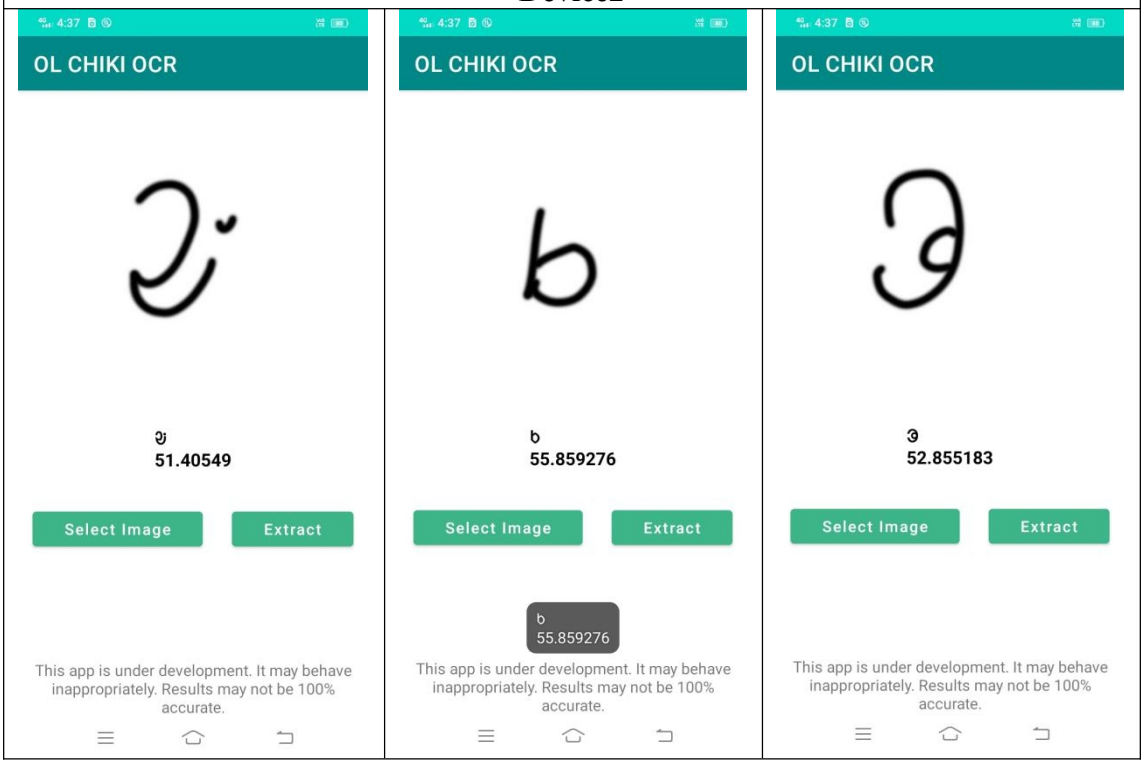
The test report is with about 50% accuracy. Screenshot of various devices(Smartphone) is attached below.

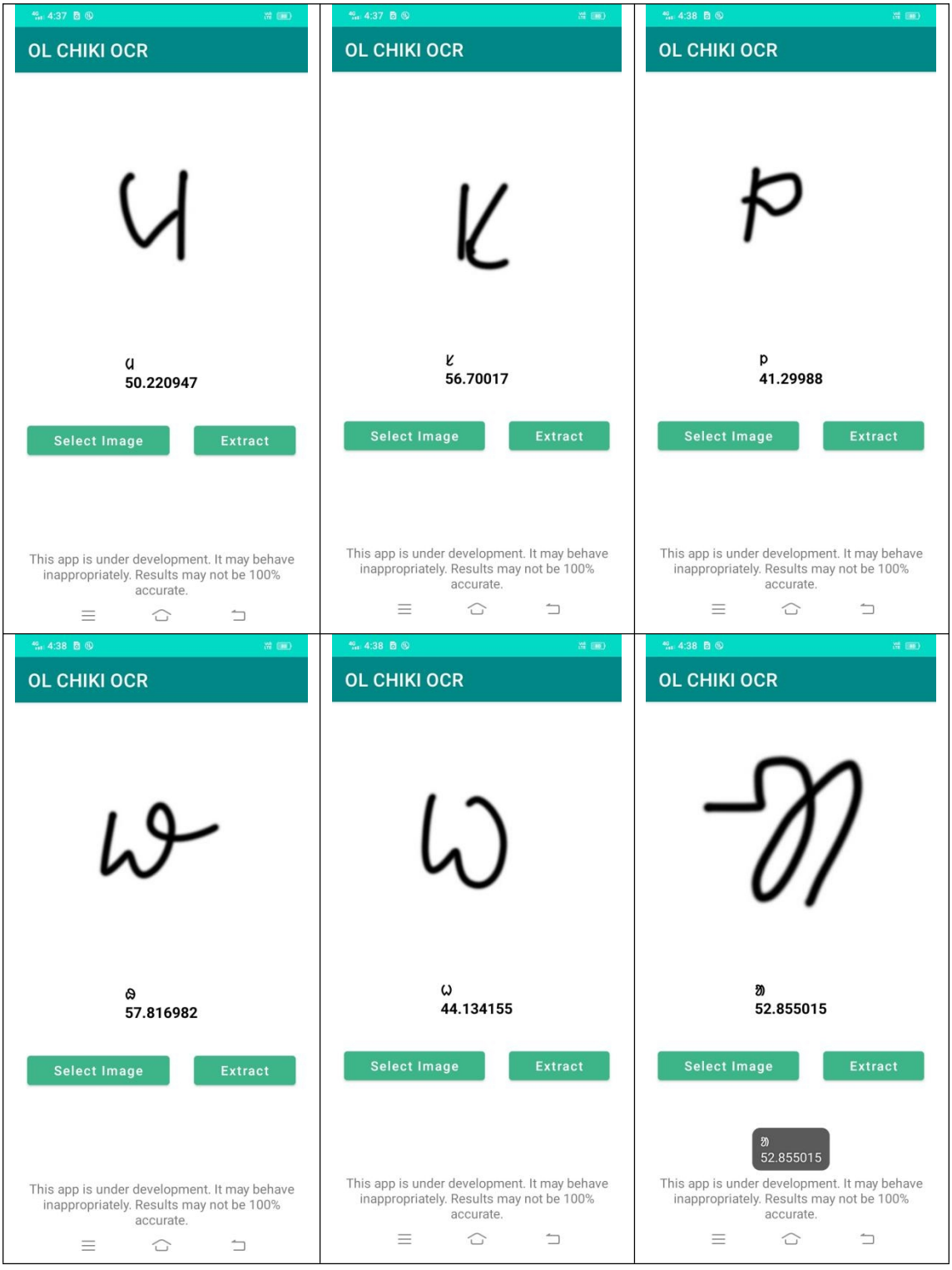
Device1



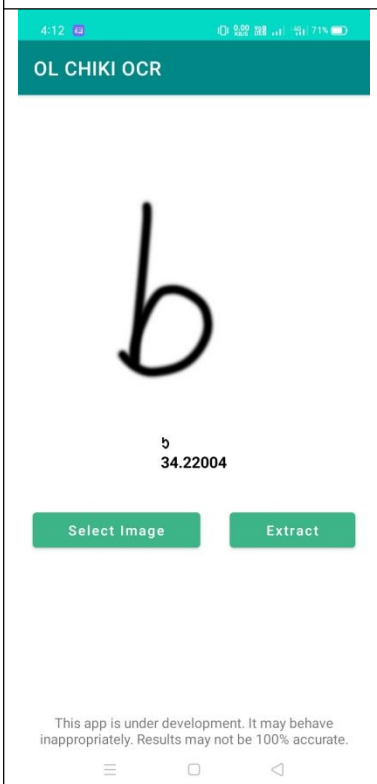

OL CHIKI OCR
ᱣ
56.70017
Select Image
Extract
This app is under development. It may behave inappropriately. Results may not be 100% accurate.
OL CHIKI OCR
ᱚ
41.986004
Select Image
Extract
This app is under development. It may behave inappropriately. Results may not be 100% accurate.
OL CHIKI OCR
ᱡ
50.052704
Select Image
Extract
This app is under development. It may behave inappropriately. Results may not be 100% accurate.

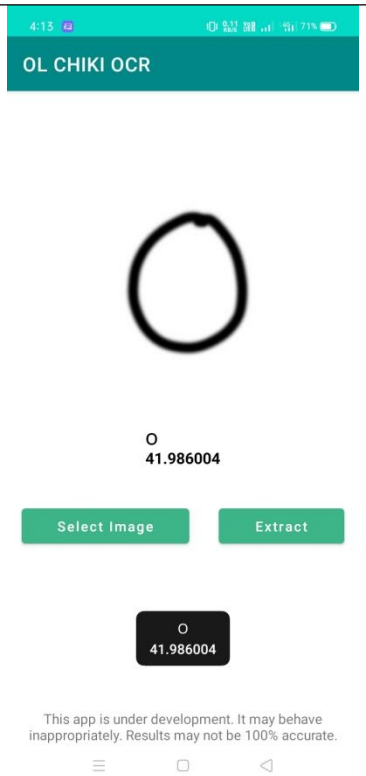
Device2

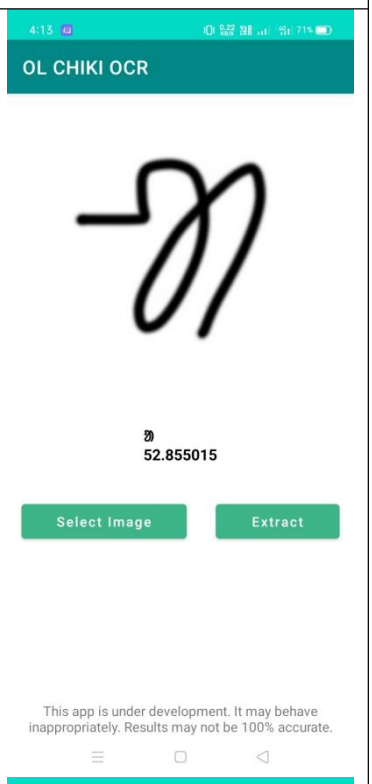

OL CHIKI OCR
ᱡ
51.40549
Select Image
Extract
This app is under development. It may behave inappropriately. Results may not be 100% accurate.
OL CHIKI OCR
ᱦ
55.859276
Select Image
Extract
ᱦ
55.859276
This app is under development. It may behave inappropriately. Results may not be 100% accurate.
OL CHIKI OCR
ᱫ
52.855183
Select Image
Extract
This app is under development. It may behave inappropriately. Results may not be 100% accurate.

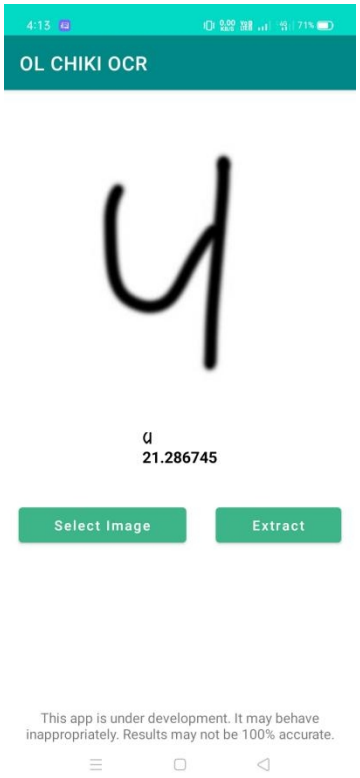

OL CHIKI OCR
ᱩ
50.220947
Select Image
Extract
This app is under development. It may behave inappropriately. Results may not be 100% accurate.
OL CHIKI OCR
ᱣ
56.70017
Select Image
Extract
This app is under development. It may behave inappropriately. Results may not be 100% accurate.
OL CHIKI OCR
ᱯ
41.29988
Select Image
Extract
This app is under development. It may behave inappropriately. Results may not be 100% accurate.
OL CHIKI OCR
ᱥ
57.816982
Select Image
Extract
This app is under development. It may behave inappropriately. Results may not be 100% accurate.
OL CHIKI OCR
ᱡ
44.134155
Select Image
Extract
This app is under development. It may behave inappropriately. Results may not be 100% accurate.
OL CHIKI OCR
ᱷ
52.855015
Select Image
Extract
ᱷ
52.855015
This app is under development. It may behave inappropriately. Results may not be 100% accurate.

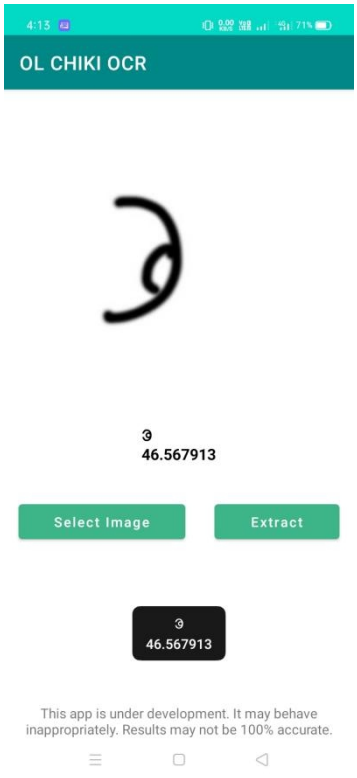
Device 3


4:12
OL CHIKI OCR
ᱦ
34.22004
Select Image
Extract
This app is under development. It may behave inappropriately. Results may not be 100% accurate.

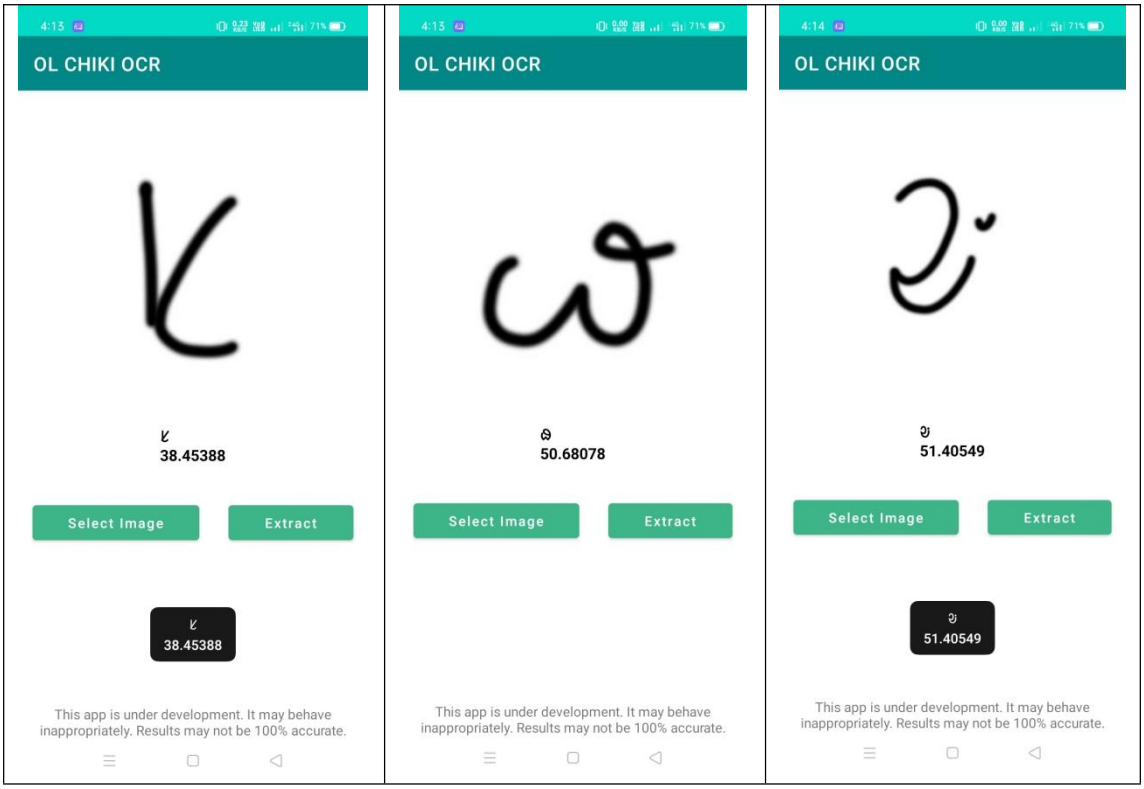

4:13
OL CHIKI OCR
ᱚ
41.986004
Select Image
Extract
ᱚ
41.986004
This app is under development. It may behave inappropriately. Results may not be 100% accurate.


4:13
OL CHIKI OCR
ᱲ
52.855015
Select Image
Extract
This app is under development. It may behave inappropriately. Results may not be 100% accurate.

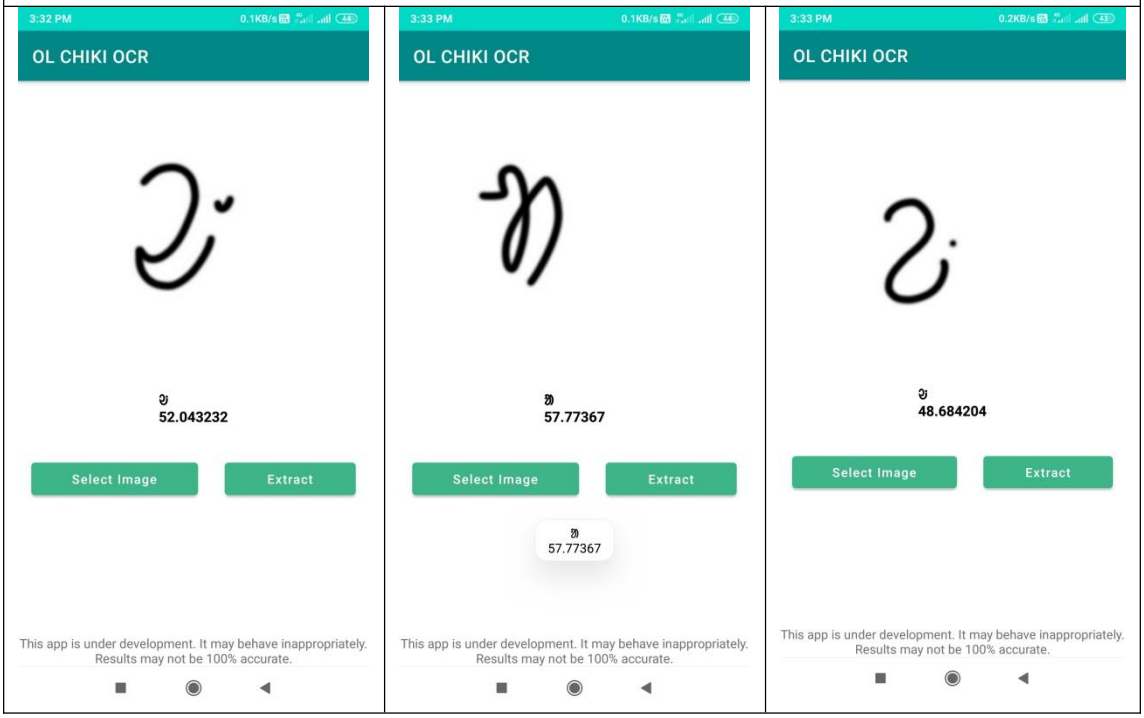

4:13
OL CHIKI OCR
ᱩ
21.286745
Select Image
Extract
This app is under development. It may behave inappropriately. Results may not be 100% accurate.

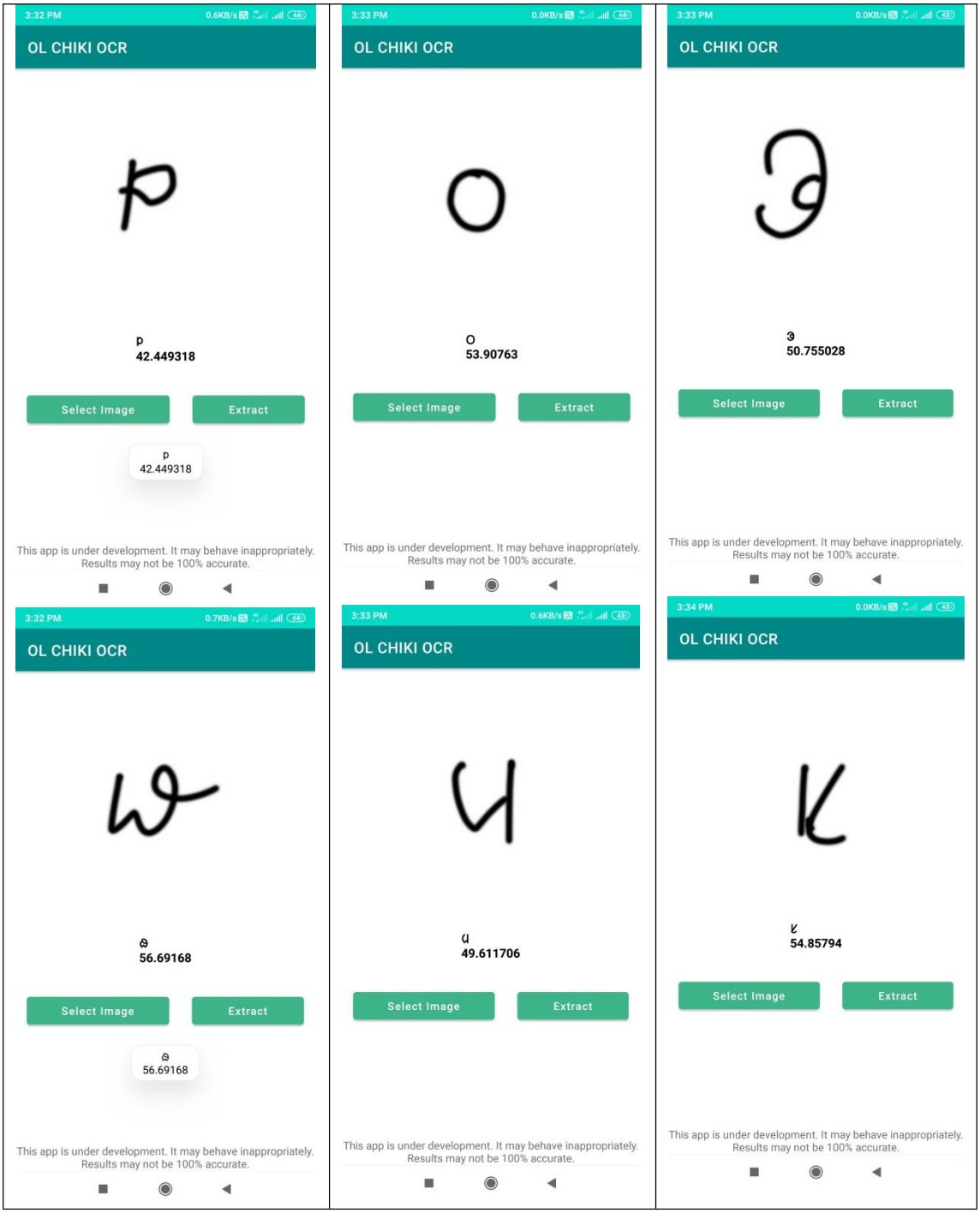

4:13
OL CHIKI OCR
ᱣ
46.567913
Select Image
Extract
ᱣ
46.567913
This app is under development. It may behave inappropriately. Results may not be 100% accurate.


4:14
OL CHIKI OCR
ᱯ
26.262686
Select Image
Extract
This app is under development. It may behave inappropriately. Results may not be 100% accurate.

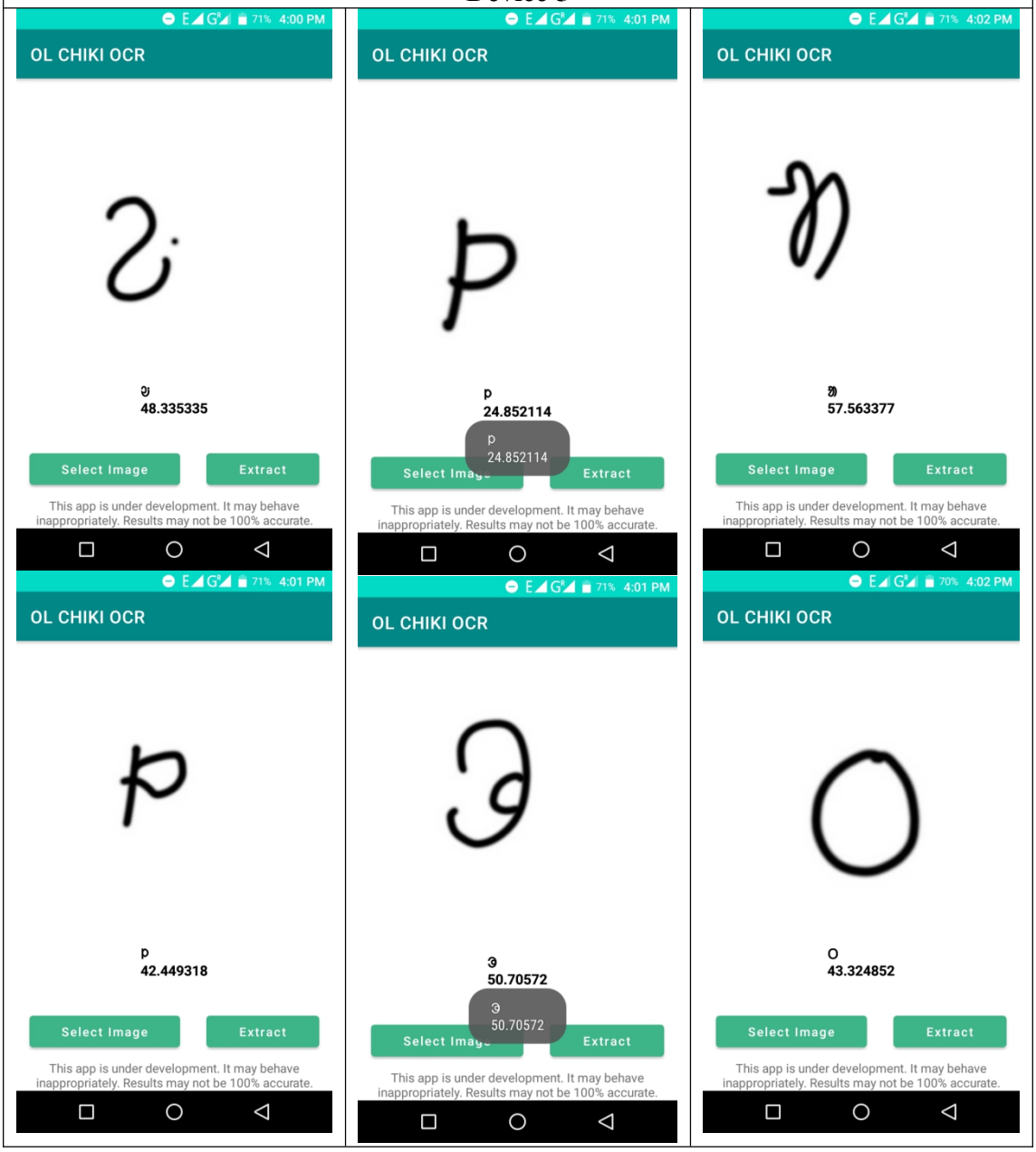

OL CHIKI OCR
ᱣ
38.45388
Select Image
Extract
ᱣ
38.45388
This app is under development. It may behave inappropriately. Results may not be 100% accurate.
OL CHIKI OCR
ᱥ
50.68078
Select Image
Extract
This app is under development. It may behave inappropriately. Results may not be 100% accurate.
OL CHIKI OCR
ᱩ
51.40549
Select Image
Extract
ᱩ
51.40549
This app is under development. It may behave inappropriately. Results may not be 100% accurate.

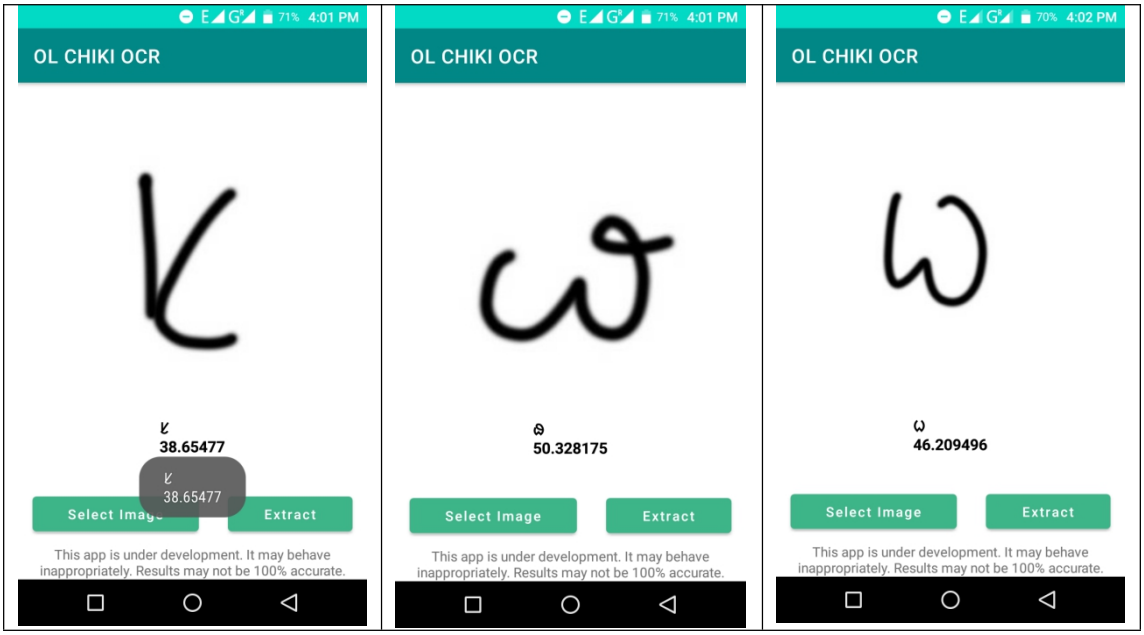
Device 4


OL CHIKI OCR
ᱩ
52.043232
Select Image
Extract
This app is under development. It may behave inappropriately. Results may not be 100% accurate.
OL CHIKI OCR
ᱦ
57.77367
Select Image
Extract
ᱦ
57.77367
This app is under development. It may behave inappropriately. Results may not be 100% accurate.
OL CHIKI OCR
ᱩ
48.684204
Select Image
Extract
This app is under development. It may behave inappropriately. Results may not be 100% accurate.

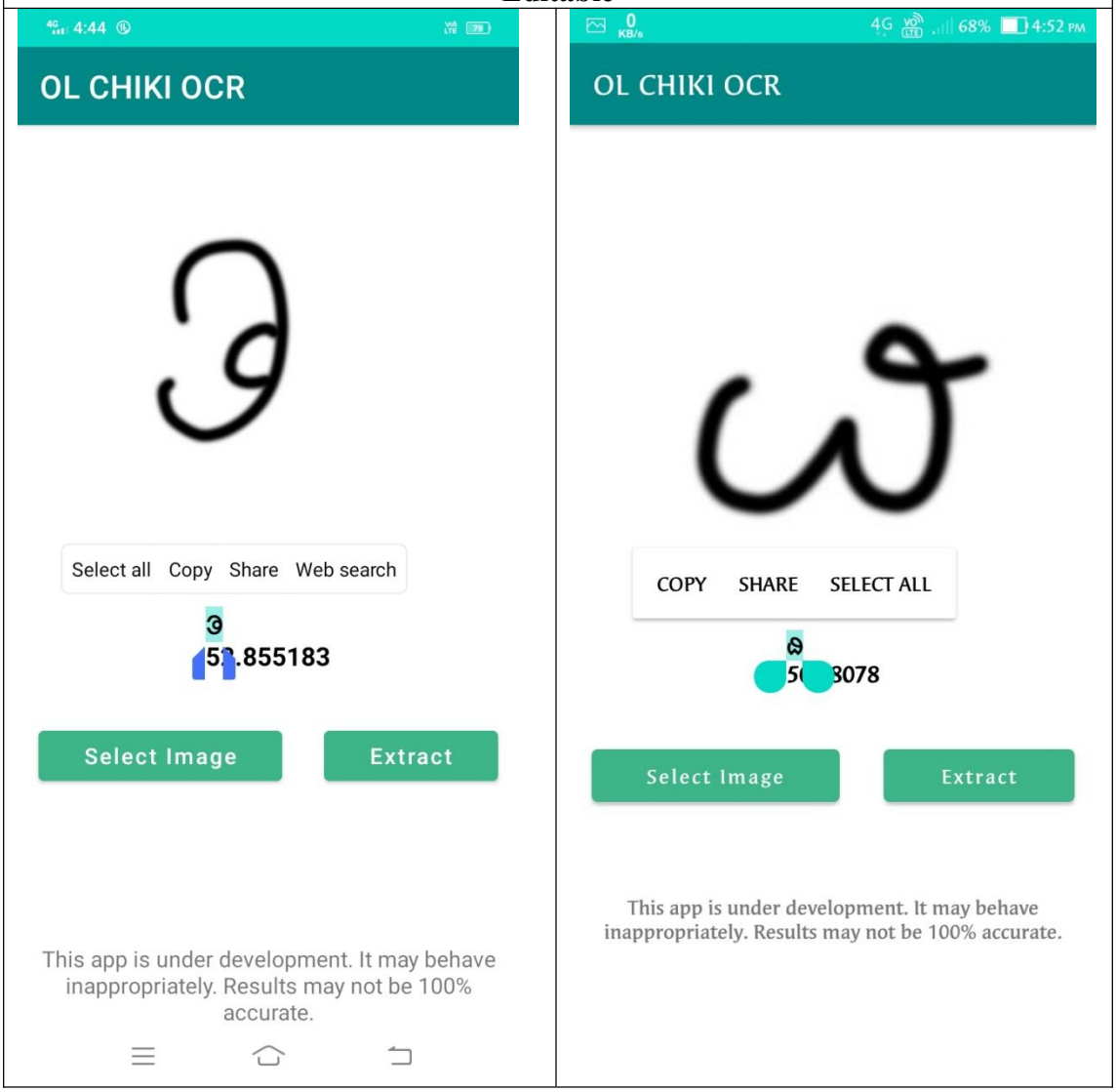

3:32 PM
0.6KB/s
OL CHIKI OCR
ᱯ
ᱯ
42.449318
Select Image
Extract
ᱯ
42.449318
This app is under development. It may behave inappropriately.
Results may not be 100% accurate.
3:33 PM
0.0KB/s
OL CHIKI OCR
ᱚ
ᱚ
53.90763
Select Image
Extract
This app is under development. It may behave inappropriately.
Results may not be 100% accurate.
3:33 PM
0.0KB/s
OL CHIKI OCR
ᱭ
ᱭ
50.755028
Select Image
Extract
This app is under development. It may behave inappropriately.
Results may not be 100% accurate.
3:32 PM
0.7KB/s
OL CHIKI OCR
ᱲ
ᱲ
56.69168
Select Image
Extract
ᱲ
56.69168
This app is under development. It may behave inappropriately.
Results may not be 100% accurate.
3:33 PM
0.6KB/s
OL CHIKI OCR
ᱩ
ᱩ
49.611706
Select Image
Extract
This app is under development. It may behave inappropriately.
Results may not be 100% accurate.
3:34 PM
0.0KB/s
OL CHIKI OCR
ᱣ
ᱣ
54.85794
Select Image
Extract
This app is under development. It may behave inappropriately.
Results may not be 100% accurate.

Device 5

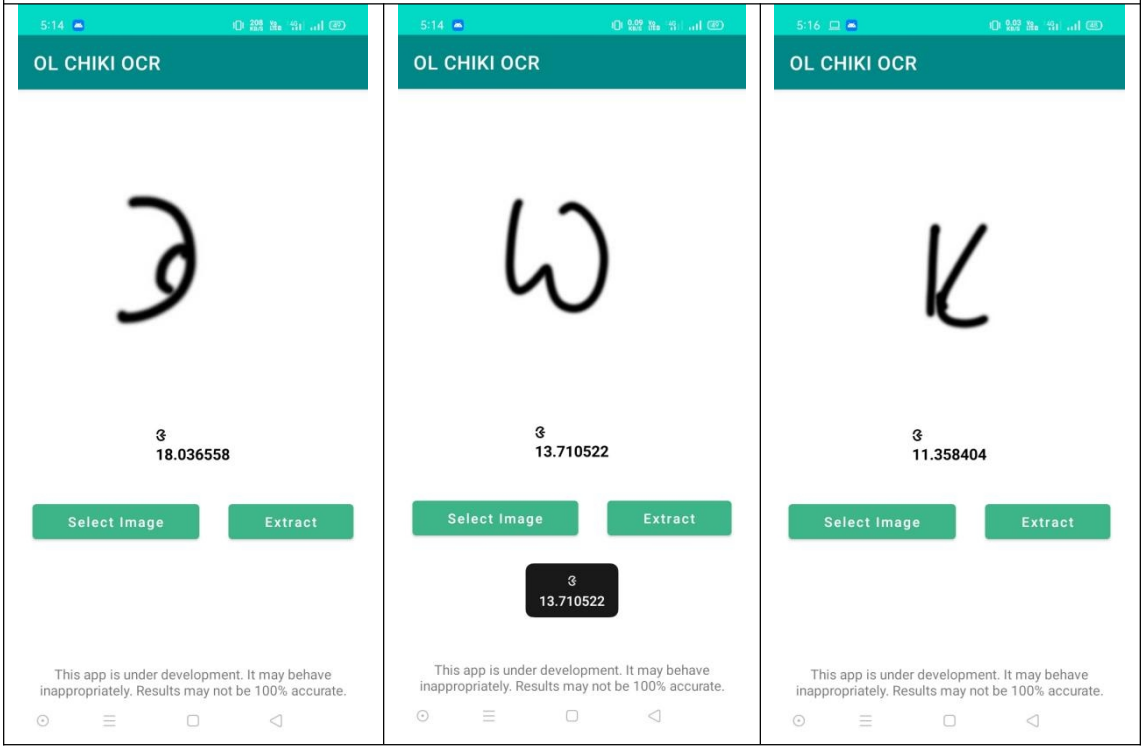

OL CHIKI OCR
48.335335
Select Image
Extract
This app is under development. It may behave inappropriately. Results may not be 100% accurate.
OL CHIKI OCR
24.852114
Select Image
Extract
This app is under development. It may behave inappropriately. Results may not be 100% accurate.
OL CHIKI OCR
57.563377
Select Image
Extract
This app is under development. It may behave inappropriately. Results may not be 100% accurate.
OL CHIKI OCR
42.449318
Select Image
Extract
This app is under development. It may behave inappropriately. Results may not be 100% accurate.
OL CHIKI OCR
50.70572
Select Image
Extract
This app is under development. It may behave inappropriately. Results may not be 100% accurate.
OL CHIKI OCR
43.324852
Select Image
Extract
This app is under development. It may behave inappropriately. Results may not be 100% accurate.


OL CHIKI OCR
38.65477
Select Image
Extract
This app is under development. It may behave inappropriately. Results may not be 100% accurate.
OL CHIKI OCR
50.328175
Select Image
Extract
This app is under development. It may behave inappropriately. Results may not be 100% accurate.
OL CHIKI OCR
46.209496
Select Image
Extract
This app is under development. It may behave inappropriately. Results may not be 100% accurate.

Editable


OL CHIKI OCR
Select all
Copy
Share
Web search
Select Image
Extract
This app is under development. It may behave inappropriately. Results may not be 100% accurate.
OL CHIKI OCR
COPY
SHARE
SELECT ALL
Select Image
Extract
This app is under development. It may behave inappropriately. Results may not be 100% accurate.

Inaccurate Results



Test Report

Different devices gave different accuracy for the same set of images containing hand written letter of **Ol Chiki** Script. Some of the devices recognized the letter with low accuracy while others with high accuracy. Some devices predicted the letters wrong with very low accuracy.

Accuracy Table

Letter	Device 1	Device 2	Device 3	Device 4	Device 5
ᱚ	51.4	51.4	51.4	52.0	48.3
ᱯ	41.2	41.2	26.2	42.4	24.8
ᱥ	57.8	57.8	50.6	56.6	50.3
ᱷ	59.2	52.8	52.8	57.7	57.5
ᱜ	46.5	52.8	46.5	50.7	50.7
ᱴ	56.7	56.7	38.4	54.8	38.6
ᱚ	41.9	40.0	41.9	53.7	43.3

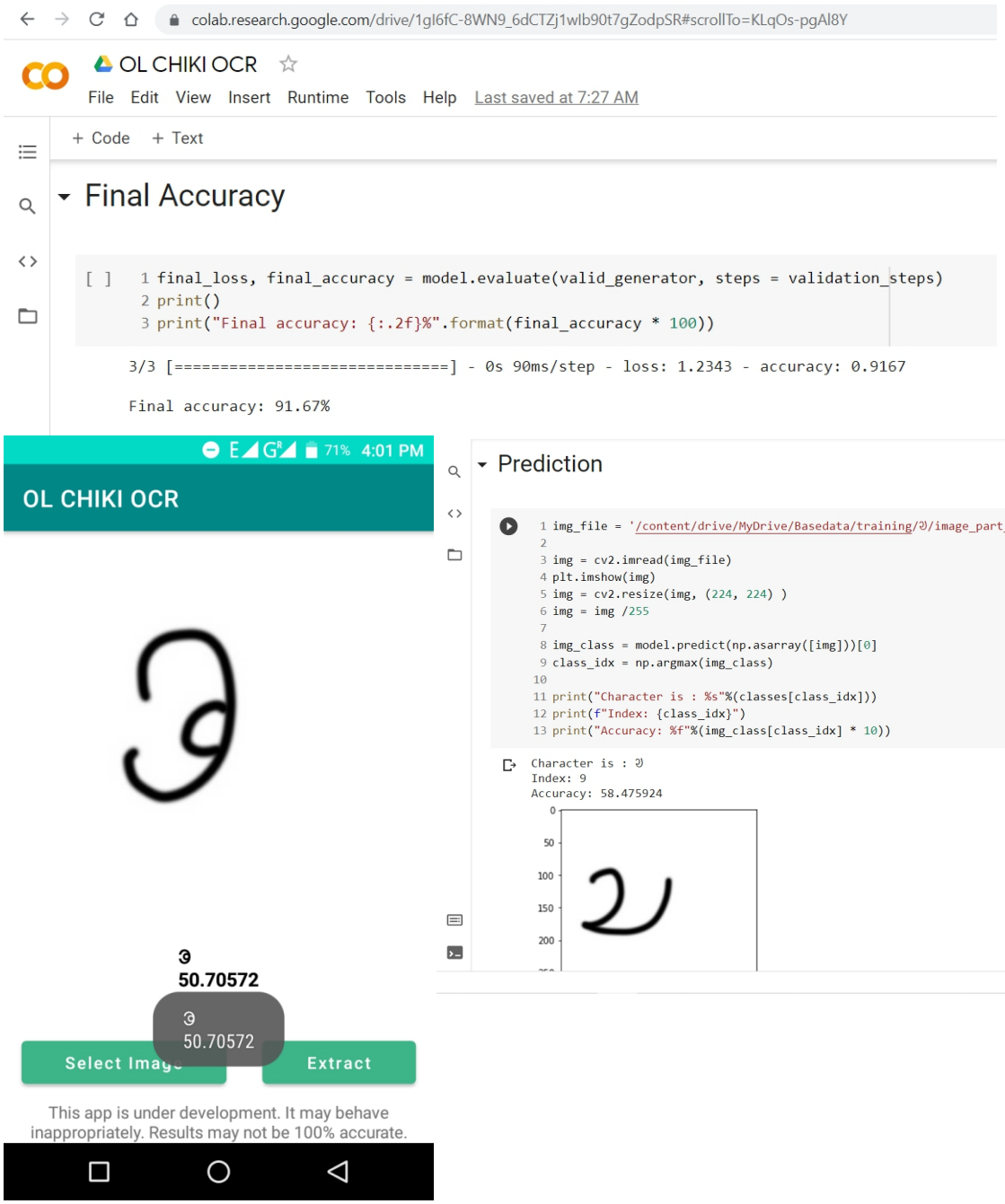
Chapter 5

Conclusion

Conclusion

Currently the accuracy of the recognition is around 50%. The trained model has acquired an accuracy of around 90%. The prediction accuracy of model is around 50% and the prediction after model implementation is also around 50%.

Currently it recognizes only a single letter.Images with more than one letter may give inaccurate results.



Limitations

There are various limitations of using **Olchiki OCR**.
Some of them are:

- It may not be 100% accurate.
- The image needs to be clear enough to extract text.
- Variation in image background may give different results.
- Currently the app recognizes only a single letter.

Future Scope

The **Olchiki OCR** android application can be enhanced in the future in different kinds of ways such as:

- Reading handwriting is a very difficult task considering the diversities that exist in ordinary penmanship. However, progress is being made.
- Recognition speed and accuracy can be increased using more no of dataset and more training.
- In future the **Olchiki OCR** will extract text from *words* and *sentences* written in *Ol Chiki* script.

Chapter 6

Bibliography

List of references :

1. **Google Drive** https://www.google.com/drive/
2. **Google Colaboratory** https://colab.research.google.com/
3. **TensorFlow** https://www.tensorflow.org/
4. **Keras** https://keras.io/
5. **Android** https://www.android.com/
6. **Android Studio** https://developer.android.com/studio
7. **GeeksforGeeks** https://www.geeksforgeeks.org/
8. **Tutorialspoint** https://www.tutorialspoint.com/
9. **Javapoint** https://www.javatpoint.com/
10. **Medium** https://medium.com/
11. **ResearchGate** https://www.researchgate.net/
12. **GitHub** https://github.com/
13. **Wikipedia** https://en.wikipedia.org/

Thank You