DEVANAGARI SCRIPT HANDWRITTING RECOGNITION APPLICATION

PROJECT WORK

Submitted in partial fulfillment of the requirement for the award of the degree of

Bachelors of Technology

In

Computer Science & Engineering

Submitted By

Milind Panwar [20SCSE1010346]

Under The Supervision of

Mr.Bibhas Kumar Rana

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING GALGOTIAS UNIVERSITY, GREATER NOIDA INDIA OCTOBER, 2021

DECLARATION

I hereby declare that this Project Report titled “Devanagari Script Handwriting Recognition Application” submitted to the “Department of Bachelor of Engineering in Computer Science & Engineering”.

It is a record of original work done by me under the guidance of Mr. Bibhas Kumar Rana.

The information and data given in the reports is authentic to the best ofmy knowledge. This project Report is not submitted to any other university or institute for the award of any degree, diploma, or published any time before.

Milind Panwar 20SCSE1010346

ACKNOWLEDGEMENT

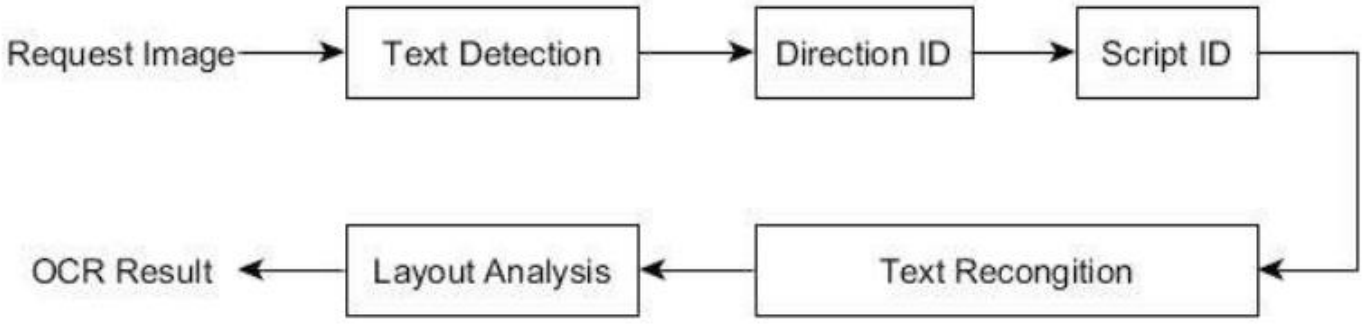
I would like to extend our gratitude to my mentor Mr. Bibhas Kumar Rana. who gave me suggestions and guided me with his ultimate knowledge and valuable time throughout the research.

I. ABSTRACT

A system for recognition of handwritten Devanagari characters has been presented for Indian writing systems. A handwritten character is represented as a sequence of strokes whose features are extracted and classified. Devanagari script is widely used in the Indian subcontinent in several major languages such as Hindi, Sanskrit, Marathi and Nepali. Recognition of unconstrained (Handwritten) Devanagari character is more complex due to shape of constituent strokes. Hence character recognition has been an active area of research till now and it continues to be a challenging research topic due to its diverse applicable environment. An android application is created as an interface to take input and show result, while The recognition of characters are done by Google Vision API.

We already have many applications that recognizes the English Handwritten script. As Hindi is Our Mothertongue and Sanskrit is Father of Many Languages, An application that recognizes the Devnagari Script and that can further be used for searching and translation purpose must exist. Devnagari Script is use to write Hindi, Sanskrit, Marathi, Nepali, Dogri, Sindhi etc. It also plays an important role in the development of manuscripts and literatures. the increase in usage of handheld devices which accept handwritten data as input created a demand for application which analyze and recognize data efficiently. Due to the popularity of digital device, in this using Smartphone as input device. Input image is drawn on Smartphone. Feature extraction of input image is done by android technology. Using that features android with machine learning recognizes the word.

DIAGRAM



Word	Correctly recognized	Misrecognized	%Recognition
50	48	2	96.00%
100	94	6	94.00%

1. LIST OF TABLES

S.NO.	PARTICULARS
1	INTRODUCTION
2	DIAGRAM
3	TOOLS AND TECHNOLOGY USED
4	RESULT AND OUTPUT
5	CONCLUSION
6	LITERARY SURVEY

2. TABLE OF CONTENT

TITLE PAGE NO.

II. INTRODUCTION

OCR is a technology that allows users to convert various documents such as scanned documents, a PDF file or images into soft copies that can be edited, searched, reproduced and easily transported. Previously, the character recognition method was limited to desktop scanners but with the advent of technology and portable computer devices like mobile phones, iPhone etc. New research methods have emerged, in which cell phones are the most widely used electronic device, eliminating the need for big machines such as scanners, desktops and laptops. Although OCR mobile packages are available on the market with a high degree of accuracy in detecting handwritten text in European and other scripts, but for Indian texts few applications are available with proper accuracy. Here, OCR based mobile systems are developed for Devanagari Script Recognizing using the Vision API. Devanagari is an alphabetic text, developed into Sanskrit but later adapted to many other languages such as Nepali, Marathi, Hindi, Konkani etc. Devanagari text has 14 vowels and 34 simple consonants. A horizontal line is drawn on top of all the letters called the headline or Shiro Rekha. Almost all Indian scripts are based on Brahmi script. Brahmi script is a phonographic writing system, in which symbols are directly related to the phonographs of the written language. A major challenge in online devanagari script is to create a system that can distinguish between variations in writing the same stroke and small variations of the same characters in the script. Many researchers have worked on a system that detects devanagari handwriting using Support Vector Machine or Particle Swarm Optimization but Vision API have high accuracy, cost effective and also supported on a portable device

III. TOOLS AND TECHNOLOGY USED

i. Kotlin :

Kotlin is a general purpose, free, open source, statically typed pragmatic programming language initially designed for the Java Virtual Machine and Android that combines object-oriented and functional programming features.

ii. Google Cloud Vision API

Google Cloud Vision API Use machine learning to understand your images with industry-leading prediction accuracy.

1) Data Collection
2) Feature Extraction

1. **Data Collection** : In the proposed method Android based Smartphone is used for taking the input image of handwriting.
2. **Feature Extraction** : Google Cloud Vision OCR is part of the Google cloud vision API to extract text from images.

iii. Android Studio :

Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on IntelliJ IDEA . Using android studio for creating an android app, which will provide the lnterface for taking input image of handwriting and displaying the detected character.

IV. RESULT AND OUTPUT

- As such there is no simple, easy and cost effective devanagari script recognition system exists for online recognition of handwritten characters with higher accuracy. I tried to develop an application using kotlin programming language and android studio as IDE with Google Vision API, which recognises the handwritten script having nearly similar accuracy as the currently proposed systems but with simple and cost effective way.
- We can use our android mobile's camera to detect the written text and that can be further used for translating and copying the text present in image.

V. CODE

```
package com.dorvis.textrecognitionandroid;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.Manifest;
import android.content.pm.PackageManager;
import android.support.annotation.NonNull;
import android.support.v4.app.ActivityCompat;
import android.util.Log;
import android.util.SparseArray;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.widget.TextView;
import com.google.android.gms.vision.CameraSource;
import com.google.android.gms.vision.Detector;
import com.google.android.gms.vision.text.TextBlock;
import com.google.android.gms.vision.text.TextRecognizer;

import java.io.IOException;
```

```
public class MainActivity extends AppCompatActivity {

    SurfaceView mCameraView;
    TextView mTextView;
    CameraSource mCameraSource;

    private static final String TAG = "MainActivity";
    private static final int requestPermissionID = 101;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mCameraView = findViewById(R.id.surfaceView);
        mTextView = findViewById(R.id.text_view);

        startCameraSource();
    }

    @Override
```

```
   public void onRequestPermissionsResult(int requestCode, @NonNull
String[] permissions, @NonNull int[] grantResults) {
        if (requestCode != requestPermissionID) {
            Log.d(TAG, "Got unexpected permission result: " +
requestCode);
            super.onRequestPermissionsResult(requestCode, permissions,
grantResults);
            return;
        }


        if (grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
            try {
                if (ActivityCompat.checkSelfPermission(this,
Manifest.permission.CAMERA) !=
PackageManager.PERMISSION_GRANTED) {
                    return;
                }
                mCameraSource.start(mCameraView.getHolder());
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
```

```
    private void startCameraSource() {


        //Create the TextRecognizer
        final TextRecognizer textRecognizer = new
TextRecognizer.Builder(getApplicationContext()).build();


        if (!textRecognizer.isOperational()) {
            Log.w(TAG, "Detector dependencies not loaded yet");
        } else {


            //Initialize camerasource to use high resolution and set Autofocus
on.
            mCameraSource = new
CameraSource.Builder(getApplicationContext(), textRecognizer)
                    .setFacing(CameraSource.CAMERA_FACING_BACK)
                    .setRequestedPreviewSize(1280, 1024)
                    .setAutoFocusEnabled(true)
                    .setRequestedFps(2.0f)
                    .build();


            /**
             * Add call back to SurfaceView and check if camera permission
is granted.
```

```
         * If permission is granted we can start our cameraSource and pass it to surfaceView
         */
        mCameraView.getHolder().addCallback(new SurfaceHolder.Callback() {
            @Override
            public void surfaceCreated(SurfaceHolder holder) {
                try {


                    if (ActivityCompat.checkSelfPermission(getApplicationContext(),
                            Manifest.permission.CAMERA) != PackageManager.PERMISSION_GRANTED) {


                        ActivityCompat.requestPermissions(MainActivity.this,
                                new String[]{Manifest.permission.CAMERA},
                                requestPermissionID);
                        return;
                    }
                    mCameraSource.start(mCameraView.getHolder());
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
```

```
            @Override
            public void surfaceChanged(SurfaceHolder holder, int format,
int width, int height) {
            }

            @Override
            public void surfaceDestroyed(SurfaceHolder holder) {
                mCameraSource.stop();
            }
        });

        //Set the TextRecognizer's Processor.
        textRecognizer.setProcessor(new
Detector.Processor<TextBlock>() {
            @Override
            public void release() {
            }

            /**
             * Detect all the text from camera using TextBlock and the
values into a stringBuilder
             * which will then be set to the textView.
             * */
```

```
        @Override
        public void receiveDetections(Detector.Detections<TextBlock> detections) {
            final SparseArray<TextBlock> items = detections.getDetectedItems();
            if (items.size() != 0 ){


                mTextView.post(new Runnable() {
                    @Override
                    public void run() {
                        StringBuilder stringBuilder = new StringBuilder();
                        for(int i=0;i<items.size();i++){
                            TextBlock item = items.valueAt(i);
                            stringBuilder.append(item.getValue());
                            stringBuilder.append("\n");
                        }
                        mTextView.setText(stringBuilder.toString());
                    }
                });
            }
        }
    });
}}}
```

VI. CONCLUSION AND FUTURE SCOPE

As such there is no simple, easy and cost effective devanagari script recognition system exists for online recognition of handwritten characters with higher accuracy. I tried to develop an application using kotlin programming language and android studio as IDE with Google Vision API, which recognises the handwritten script having nearly similar accuracy as the currently proposed systems but with simple and cost effective way.

VII. LITERARY SURVEY

H. Swethalakshmi, Anitha Jayaraman, V. Srinivasa Chakravarthy, C. Chandra Sekhar [1] worked on the Devanagari script recognizer and their Devanagari script detector was trained in stroke data collected from 90 users and tested on data from 10 users . The total number of models used for training was 21780 and for testing was 2420. The total number of classes considered in the Devanagari script was 91. They came to the conclusion that the performance of the character recognition depends on the accuracy of the side recognition. The results obtained for Devanagari character recognition indicated that reliable classification was possible using SVMs. (SVM is for mapping the input data of a higher dimension feature unrelated to the input area and decides to split a large plane with a large limit between the two sections in the feature space.) Prashant M. Kakde, S.M. Gulhane, [2] Tried to create a system that uses a new way to fine-tune particles. Use PSO and SVM to recognize text from real-time installations using a combination of Android, PHP and MATLAB and reach a point where the PSO method provides much better accuracy compared to SVM. The accuracy of the real-time system generated for this task is about 90%. (PSO is a human-based search method. PSO is a collaborative research method, as it makes few or no assumptions about the problem but will search for the largest gaps in the solution.) Hanmhunga, Madasu et al. [3] used the modified membership function to represent unambiguous sets from sample elements and then applied reinforcement training to structural parameters resulting in a 25-fold improvement in the mixing speed. In the analysis of documents, when computer time was a major factor, this study was predicted to be useful. The total recognition rate in rough classification was found to be 90.65%. M. Yadav, R. Kr Purwar, and A. Jain [4] examined many concessions and combinations of convolutional layers. They used two test analytics data that achieved a high accuracy of 97.95% but the Convolutional neural network has high computational costs and a large storage space, so they suggested a better approach could be developed to develop a cost-effective character recognition system..

VIII. REFERENCES

[1] *H. Swethalakshmi, Anitha Jayaraman, V. Srinivasa Chakravarthy, C. Chandra Sekhar. Online Handwritten Character Recognition of Devanagari and Telugu Characters using Support Vector Machines. Tenth International Workshop on Frontiers in Handwriting Recognition, Université de Rennes 1, Oct 2006, La Baule (France). ffinria-00104402f*

[2] Prashant M. Kakde, S.M. Gulhane, A Comparative Analysis of Particle Swarm Optimization and Support Vector Machines for Devnagri Character Recognition: An Android Application Procedia Computer Science, Volume 79,2016, Pages 337-343, ISSN 1877-0509,

[3] Hanmandlu, Madasu et al. "Fuzzy Model Based Recognition of Handwritten Hindi Characters." 9th Biennial Conference of the Australian Pattern Recognition Society on Digital Image Computing Techniques and Applications (DICTA 2007) *(2007): 454-461.*

[4] M. Yadav, R. Kr Purwar, y A. Jain, Design of CNN architecture for Hindi Characters, ADCAIJ, vol. 7, n.º 3, pp. 47–62, sep. 2018.