Journal of Information Systems Engineering and Management
2025, 10(41s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

A Framework for Character Recognition APP Using ML Kit

Dr. Deepika Kongara[1], Shivani Krishnama[2]
[1] *Assistant Professor, Department of Information Technology, Kakatiya Institute of Technology and Science, Warangal, Telangana, India.*
[2] *Department of Information Technology, Kakatiya Institute of Technology and Science, Warangal, Telangana, India*

ARTICLE INFO

Received: 31 Dec 2024

Revised: 20 Feb 2025

Accepted: 28 Feb 2025

ABSTRACT

Character recognition applications are pivotal in enabling real-time translation and digitization of printed and handwritten text. Text recognition can be implemented using a variety of technologies found in the field of software development, but here for Android mobile development, the ML Kit is used. The goal is to create an Android character recognition app, especially for Devanagari script (with language converter software included as a feature), using the ML kit without Firebase that will make recognizing, learning, and language translation easier and will promote stress-free communication. Using ML Kit is a boon for developers without extensive knowledge of machine learning, as it simplifies the integration of complex ML features and saves significant time in learning and implementation. This app has the ability to recognise in real time as well as storage-based recognition on both handwritten and printed scripts. The app incorporates real-time recognition and offline translation features, supporting recognition in six languages and translating text into 59 languages. This app will perform more effectively than other existing programmes since it will use optimized code for the translation and recognition process. The major goal is to have the software operate even when it is not connected to the internet.

Keywords: *ML Kit, Android Studio, Translation, Devanagari, Dependencies, Recognition, Debugging, Chinese, Kotlin, Xml, Google lens.*

1.INTRODUCTION

The popularity of character recognition software has grown over the past several years, along with machine learning and artificial intelligence. Numerous industries, including education, healthcare, the IT sector, and finance, could benefit from the use of this technology, which has the capability of recognizing both printed and handwritten text from pictures. Character recognition technology is a crucial tool for intercultural communication since it can convert recognised characters into several languages. In simpler words, "character recognition" is a picture-to-text conversion that generally uses a ML model or libraries. The main goal of this work is to develop a character recognition app (along with a translator) using ML Kit, especially for Devanagari script.

Although machine learning is quickly taking on a significant role in mobile app development, it isn't the simplest feature to incorporate. To take advantage of ML, one usually needs a solid grasp of neural networks and data analysis, as well as the time and resources required to collect enough data, train your ML models, and then optimize those models for mobile. Mobile applications and many other things around us today use some form of machine learning, which has already begun to change how we live. Google has launched ML Kit to make implementing machine learning easier for Android app developers. It supports several pre-built machine learning models, including text recognition, face detection, image labeling, translation, and more. Also, it supports implementing new models using TensorFlow Lite in iOS, Android, and web apps. The main goal of this work is to incorporate ML onto mobile devices using ML kit.

1.1 ML Kit

ML Kit is a machine learning framework for creating mobile apps. It is open source and free to use. It offers pre-built and customized models for typical use cases, including image and text recognition, pose detection, labeling and more. ML Kit is compatible with common mobile development platforms, including Android and iOS, and it supports a

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2025, 10(41s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

broad range of device architectures. It also provides developers with simple APIs and tools for easily incorporating machine learning capabilities into their projects without requiring considerable machine learning experience. ML Kit also supports on-device and cloud-based processing, allowing developers to balance performance, privacy, and scalability requirements.

1.2 Objectives

This work aims to create a useful Android application that enables users to extract and identify characters from pictures (both handwritten and printed scripts), especially Devanagari script, using ML Kit in offline mode. This application also aims to incorporate a translator that can convert the recognised characters across different languages. The main goal is to offer an open-source project that other developers interested in creating comparable apps can use as a guide or a jumping-off point.

1.3 Applications and Limitations

The character recognition app has potential uses that can benefit various individuals and industries. For language learners, the software can recognise and translate characters, which can aid in learning and comprehension. Additionally, tourists may utilize the app while traveling to read and comprehend foreign language signs, menus, and other written information. Individuals with visual impairments may also benefit from the app, as it can help them read and interpret written information more easily.

Instructors can also use the software to recognise and correct student work, especially when it comes to handwriting recognition. Also, the app can be employed to translate legal papers written in multiple languages quickly and accurately. This application may be integrated with websites to automatically translate text and graphics to various languages, which can enhance the accessibility and usability of the website.

As the software can recognise text in photos, it can be used to scan and digitize documents, which can streamline the documentation process. The app can also be used to organize photo collections, making it easier to search for specific images based on the text present in the photos.

This application is limited to recognition and translation for a select number of languages (59), as it is developed without Firebase. Developers are provided with two choices: on-device mode (offline mode) and cloud mode. The on-device mode, which is used in this application, has limitations in terms of language support and customization of machine learning models compared to the cloud mode. While cloud mode offers better performance and supports more languages, on-device mode allows for processing multiple images and handling large volumes of text locally. Offline mode is chosen here for its privacy and speed, as all processing is done locally and can be accessed remotely.

2.LITERATURE SURVEY

Character recognition has a long history that dates back to the early twentieth century, when optical character recognition (OCR) technology was first created. Since then, several OCR systems have been created, including techniques based on support vector machines, artificial neural networks, and pattern recognition. Convolutional neural networks (CNNs), which have been demonstrated to produce top results in character recognition tasks, have become more popular in recent years. A popular emerging tool for integrating machine learning models into Android apps is ML Kit, which offers a number of pre-built models for text translation and picture recognition. The performance of the text translation model in ML Kit was compared to Google Cloud Translation in one pertinent research publication by H. J. Kim et al. [1]. The study discovered that ML Kit's translation model had an average accuracy of 90.35%, making it very accurate. For the purpose of identifying handwritten Devanagari characters, Sandhya Arora et al. [2] evaluated the abilities of two well-known machine learning methods, Support Vector Machines (SVM) and Artificial Neural Networks (ANN). They compared the accuracy of these two algorithms and came to the conclusion that SVM performs better than ANN in terms of recognition rate.

U. Pal and R.B. Chaudhuri [3] gave an overview of character identification methods for Indian script. They performed a number of strategies, including neural network-based, statistical, and structural. They also draw attention to the difficulties with identification, such as the variety of writing styles and the use of diacritical markings in their study.

To recognise printed Devanagari characters, Manoj Kumar Shukla and Dr. Haider Banka [4] suggested an effective segmentation system. They integrated several image processing methods to separate out certain characters from the

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

source image. The method was then tested on a dataset of 4800 characters, and the results showed a 95.1%-character identification rate.

A research to compare the performance of Java with Kotlin in Android app development activities was carried out by Ardito et al. [5]. They discovered that Kotlin is much more productive and produces higher-quality code than Java because it is more understandable, and error-resistant. Kotlin performs better in terms of memory utilisation as well as compilation time, according to the experimental results done by them.

Malanker A. and Patel M. [6] provided a survey of techniques for recognizing handwritten Devanagari script. They highlighted the importance of segmentation and writer-independent recognition for achieving accurate character recognition rates in their paper.

Indira B. and Sudha T. [7] provided a practical method for deciphering Indian vehicle number plates. They extracted the characters from the license plate using image processing methods, and then they utilized a template matching method to identify them. Then they tested their strategy on a dataset of 50 license plates and got a 98% recognition rate.

Pooja Sharma [8] covered various techniques and approaches used for Devanagari character recognition, including preprocessing, segmentation, feature extraction, and classification. She concluded that the majority of the recognition system's faults are caused by segmentation and the rate of recognition can be raised if we recognise the entire word without segmenting it.

The study by Trier et al. [9] provided a thorough analysis of feature extraction techniques for character recognition. They further discussed different methods for extracting features from character pictures. Then they evaluated the efficiency of several feature extraction techniques and discussed the benefits and drawbacks of each method in their paper.

Bansal and Sharma [10] covered the methods for segmenting individual handwritten words in Gurmukhi script. They explored several strategies, as edge-based and histogram-based, then determined the efficiency of each strategy. They also emphasized the significance of preprocessing processes for accurate segmentation, such as noise reduction and binarization.

The segmentation of isolated and touching characters in offline handwritten Gurmukhi script recognition are covered by Kumar et al. [11]. They examined several strategies for their efficiency. They further discussed various segmentation techniques, such as vertical projection, horizontal projection, and connected component analysis, and evaluated their effectiveness in character segmentation.

Garg et al. [12] explored how to divide half-characters in handwritten Hindi language. They examined several segmentation techniques and evaluated their effectiveness in segmenting those half characters.

Hennig et al. 's [13] gave a description of ML Kit and its capabilities for mobile machine learning. The pre-built models in ML Kit, such as face identification, word recognition, and picture labeling, are discussed in their article.

Benois-Pineau et al. [14] investigated the application of ML Kit for mobile device real-time object identification. The study assessed the object identification model's performance across a range of gadgets and discovered that it was capable of real-time performance on contemporary smartphones.

The integration of ML Kit onto mobile applications is covered by N. B. Shridhar et al. [15], who also gave examples of how ML Kit may be used for a variety of machine learning tasks, such as text and picture recognition.

Dr. Mohammad Alshraideh et al. [16] explored various methodologies for recognizing handwritten and typed Arabic letters, employing different algorithms for segmentation, skeletonization, feature representation, and classification. They utilized techniques such as thinning algorithms, boundary detection, the fuzzy ISODATA algorithm, and parallel thinning, alongside both traditional machine learning and deep learning methods. Their proposed prototype

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2025, 10(41s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

achieved up to 98% accuracy in the best-case scenario and 72% in the worst-case scenario. Their approach not only improved accuracy but also addressed challenges such as script variations, ligatures, and the unique contextual understanding required for Arabic text, thereby establishing a new benchmark in OCR technology for Arabic languages.

3. METHODOLOGY

Instead of choosing a dataset, preprocessing it, cleaning dataset, experimenting with various algorithms to improve accuracy, training and testing the machine learning model or customizing ML model then implementing it into an app. One can simply choose the ML Kit. One can save a tonne of time, space, and money by using ML Kit's pre-trained character recognition models rather than having to create our own Machine Learning Model. One significant benefit of the ML kit is that programmers don't need to be experts in machine learning. Without having any prior experience with machine learning, a simple app may be created using the ML Kit.

The ML Kit is offered as a library or SDK that may be simply included into a project. To incorporate ML kit into an Android app, we must add Google's ML kit dependencies to the project's build file. After integrating the ML kit, the user has two options: customize the ML kit or continue using the pre-trained models of the ML kit. To customize, users must utilize Firebase by establishing a project on Google Firebase and enabling ML kit in the Firebase dashboard to have access to the ML kit's features and APIs. If a created model is too huge, developers can store it to the cloud and then arrange for it to be downloaded dynamically to the app. We utilized a pre-trained ML Kit model in this case. After enabling

ML kit, we must select which ML kit functionalities will be used in the app. Text Recognition, Image Labeling, Object Detection, Language Translation, Smart Reply, and other features are included in the ML kit. In our app, we employed the kit's Text Recognition and Translation features. These needed functionalities are added to the app in accordance with the standards outlined in Google's ML Kit Guides. Depending on their requirements, developers can choose between on-device and cloud APIs. We chose the on-device architecture in this app because it takes up less space, can run offline, and analyzes data quicker. It is ready to program once the ML kit features have been added. Currently, the Text Recognition feature supports 6 languages, whereas the Offline Translation option supports 59 languages.

KEY FEATURES of app are as follows-

1. Real-time Recognition.
2. Offline Based Recognition and Translation.
3. Storage based recognition.
4. Both handwritten and printed text recognition.

The user interface and the backend functionality are two main parts of this proposed app. The app's user interface, which is in charge of showing the app to the user and obtaining input from them, is created using XML and Kotlin. The user interface consists of a single display, the home page, where the user may capture a picture or choose an image from the gallery, and then the translated and recognised text is presented on the same page.

The backend component of the app is in charge of handling user inputs, extracting characters from a picture, and translating the text that has been extracted. It is developed using ML Kit and other libraries. To avoid having to build a machine learning model, ML Kit offers a character recognition pre-trained model. The translated text is then sent to the translation library, which translates it into the target language.

3.1 Working on Text Recognition

After setting up the ML kit, create an InputI mage object from a Bitmap, media. Image, Byte Buffer, byte array, or a file on the device to recognise text in an image. The Input Image object is then sent to the TextRecogniser's process Image function. Following image processing, recognition happens in the following stage. In this project, we calculated rotation values using the CameraX framework. The structure of the recognised text is organized into distinct components in the Text Recognition API. A "Block," which depicts a chunk of text in the picture, is one of these components. Each Block can have one or more "Line" components, and each Line can have one or more "Element" objects in it.

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2025, 10(41s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

The Block object contains information pertaining to the recognised text within that block. A Block object has the following properties:

1. Text: This attribute, which is kept as a string, reflects the recognised text found across the whole block.
2. Frame: A Rect object serves as the defining feature of a Block's Frame attribute. The four values that correspond to the corners of the rectangle enclosing the block are defined by a Rect object. These numbers represent the rectangle's left, right, top and bottom coordinates.
3. Corner Points: The Block object offers the Corner Points property in addition to the Frame property. The Corner Points property offers the whole set of coordinates for the four corners of the detected text box rather than just the four values of the rectangle (left, top, right, and bottom) as is the case with the Frame property. This enables more accurate spatial localisation by providing additional information about the text box's vertices.
4. Language Code Recognised: This attribute shows what language the text that was recognised was written in. The recognised language code is not a direct feature of the ML Kit Text Recognition API. However, it has language identification capabilities that let you identify the language of the recognised text by utilizing extra ML Kit APIs or outside language identification libraries.
5. Lines: A Block's sub element. A Block object in the Text Recognition API may contain one or more Line objects. A line of text within the block is represented by each Line object. These Line objects may be accessed using a Block's Lines property. One or more Element objects, which stand in for smaller portions or units of text inside the line, can be included within a Line object.

Examples of each of them are shown in descending order in the graphic below. The first portion of the image that is highlighted is a block of text made up of many lines. Lines are the next highlighted section in blue, and components are the following highlighted section in green. Figure 1 is shown below.

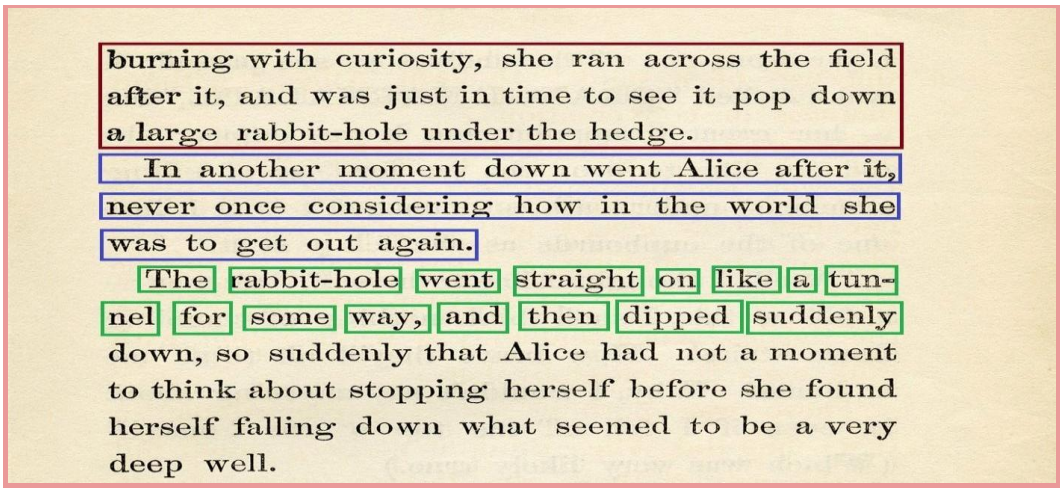

Figure 1: Text Recognition API's text structure.

3.2 Overview of Implementation Steps

The implementation involves -

1. Integration of ML Kit - Add ML Kit to user's Android Studio project and pick the character recognition model offered by ML Kit according to user choice. ML Kit at present includes character recognition models for six languages (without Firebase).
2. Translation service integration - Integrate Google's Translate API into the app to give translation services for the recognised characters or any other translation needed by the user (it does not have to be simply the recognised text).
3. User interface - Create an XML user interface for the application and implement it in Kotlin or Java.

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2025, 10(41s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

4. Loading model - Load the character recognition model after building builders for desired language and also load the translator function.
5. Error handling - Correcting mistakes that may come up while character recognition is being done. For instance, notifying the user of an error if the image is hazy or the text cannot be read correctly.
6. Testing and evaluation - Assess the application's performance in terms of accuracy, speed, and user experience by testing (manually by using the app) it on a range of photos both printed and handwritten.
7. Deployment: Upload the application to the Google Play Store or other app marketplaces.

4.EXPERMENTATION AND RESULTS

In this phase, the developed model and the ML Kit's pre-built models were integrated into the character recognition app. The app was created using the Android Studio IDE, and the Kotlin programming language was used to write the code. The app's user interface was designed to be simple and user-friendly, with an easy-to-use camera feature for capturing images of characters to be recognised. The captured images were then processed using the character recognition model, and the recognised characters were displayed on the app's screen. Additionally, the ML Kit's pre-built models for text translation were integrated into the app to enable real-time translation of the recognised characters.

4.1 Setting up Development Environment

Android Studio can be downloaded from their official website (https://developer.android.com › studio › download). After installation configure the app for further use.
Create a new project in Android Studio by adding a new project or File>New>New Project. Then pick a project name, a package name, and Kotlin as the language.

4.2 Initializing ML Kit

Every Project consists of two gradle scripts build. gradle (Module: app) and build. gradle (Project). The build. gradle (Project) file provides global project settings and configurations, such as Gradle plugins, repositories, and other options while the build. gradle (Module: app) file provides configurations for the individual module or application being produced, such as the app version, dependencies, and other parameters. As a result, dependencies must be specified in the build. gradle (Module: app) file. This is where one should include the required dependencies in the app, such as libraries or SDKs. Figure 2 below shows how and what dependencies can be added.

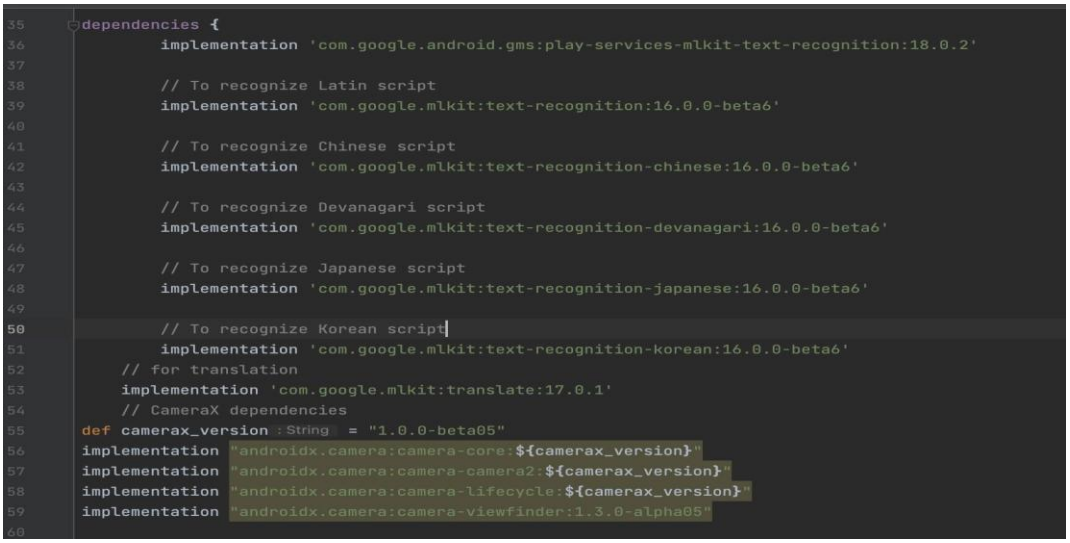

Figure 2: Adding Dependencies to Project.

Dependencies can be copied from official website developers.google.com. Users have the choice to implement their desired language character recognition. This project is more specifically for Devanagari Character Recognition but for testing all six languages are implemented here.

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2025, 10(41s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Here CameraX dependencies are also imported for permissions.

4.3 Designing a Layout

Using XML a layout for the application is created. Developers can get as creative as possible playing with icons, designs, images, transparency, location of properties as buttons, textboxes and more. Figure 3 is shown below.

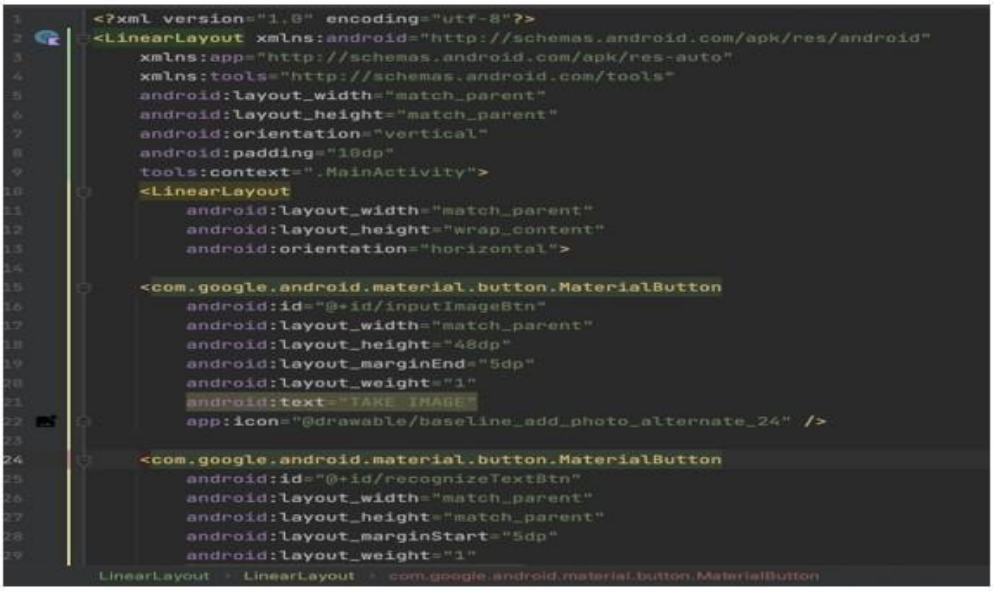

Figure 3: Sample Code for Application Layout.

4.4 Managing Permission Requests

In Mainactivity itself all of the responsibilities that are linked to the Text Recognition API, such as managing button click events, and requesting access to the device's storage are managed using kotlin. This project is done using Kotlin Language rather than Java because of its efficiency. Figure 4 below demonstrates the code for app permissions.

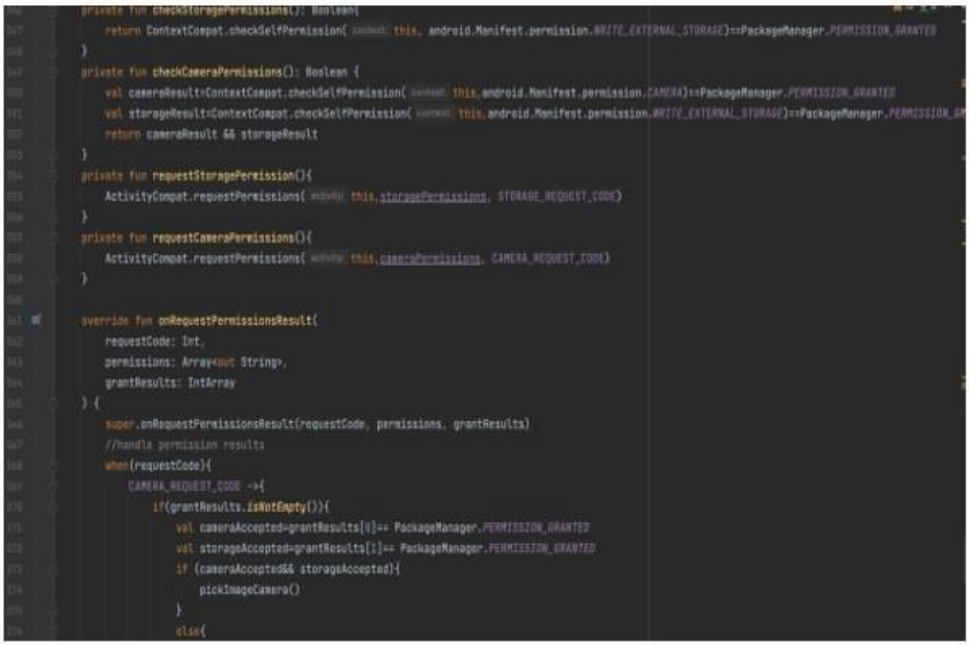

Figure 4: Code for Camera and Storage permissions.

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2025, 10(41s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Request permission to utilize the camera, internet and storage by including the following in your AndroidManifest.xml file. As in the code from Figure 5 shown below.

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
    <uses-permission android:name="android.permission.CAMERA"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <application
```

Figure 5: Code for Permissions.

4.5 Loading Recognition Model

Load the character recognising model which does the work of recognising characters. Figure 6 is shown below.

```
//this will load model
textRecognizer = TextRecognition.getClient(DevanagariTextRecognizerOptions.Builder().build())
```

Figure 6: Model Loading for Character Recognition

4.6 Text Recognition API

A picture is sent as input to the TextRecognition API. The CameraX image buffer returns an instance of an Image object containing all of the data transmitted by the camera's hardware. Following that, we pass the image instance to the API, which returns a Text object.

The Text object is separated into hierarchies as blocks, lines and elements. As a result, one Block comprises one or more Line elements, and each Line element is made up of one or more Element objects. These contain the same information.

4.7 Loading Translation Model

The next stage in the process is to provide functionality for the translator's source and destination language selections. This entails providing code methods that allow the user to select the source and destination languages for translation. After that, the code for the translation process may be added. The recognised text is subsequently passed to the translation model, which then translates it to the appropriate language. The user is subsequently presented with the translated content. This phase is critical in allowing the user to translate the recognised text from the source language to a language with which they are more familiar. The text is sent to the translation library, which uses a translation API to convert it into the required language.

Add functions for source language options as well as destination language options for the translator. Figure 7 is shown below.

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2025, 10(41s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

```
private fun sourceLanguageChoose(){
    val popupMenu=PopupMenu( context: this,sourceLanguageChooseBtn)
    for(i in languageArrayList!!.indices) {
        popupMenu.menu.add(Menu.NONE, i, i, languageArrayList!![i].languageTitle)
    }
    popupMenu.show()
    popupMenu.setOnMenuItemClickListener {menuItem->
        val position=menuItem.itemId
        sourceLanguageCode=languageArrayList!![position].languageCode
        sourceLanguageTitle=languageArrayList!![position].languageTitle
        sourceLanguageChooseBtn.text=sourceLanguageTitle
        sourceLanguageEt.hint="Enter $sourceLanguageTitle"
        Log.d(TAG, msg: "sourceLanguageChoose: sourceLanguageCode: $sourceLanguageCode")
        Log.d(TAG, msg: "sourceLanguageChoose: sourceLanguageTitle: $sourceLanguageTitle")
```

Figure 7: Code for Source Language Choosing.

In the next step add a process language model for translation. Figure 8 is shown below

```
private fun startTranslation(){
    progressDialog.setMessage("processing language model")
    progressDialog.show()
    translatorOptions=TranslatorOptions.Builder()
        .setSourceLanguage(sourceLanguageCode)
        .setTargetLanguage(targetLanguageCode)
        .build()
    translator=Translation.getClient(translatorOptions)
            val downloadConditions=DownloadConditions.Builder()
        .requireWifi()
        .build()
    translator.downloadModelIfNeeded(downloadConditions)
        .addOnSuccessListener { it: Void!
            Log.d(TAG, msg: "startTranslation: model ready,start translation...")

            progressDialog.setMessage("Translating...")
            translator.translate(sourceLanguageText)
                .addOnSuccessListener {translatedText->
                    Log.d(TAG, msg: "startTranslation: translatedText: $translatedText")

                    progressDialog.dismiss()
                    targetLanguageTv.text=translatedText

                }
                .addOnFailureListener{e->
                    progressDialog.dismiss()
                    Log.e(TAG, msg: "startTranslation:",e)
                    showToast("failed to translate due to ${e.message}")

                }
```

Figure 8: Code for Translation.

Now the project is ready for execution. Before execution try to sync the files again to avoid sync errors. Check for any possible errors and preferably provide conditional loops as if else to identify the specific errors. Test the app manually using different input images on a variety of devices and environments to ensure its functionality and usability.

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2025, 10(41s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

4.8 Debugging

Debugging in Android studio can be done in different ways-

USB debugging is the most frequent method for debugging and running an Android app. Connect the user's Android device to the computer using a USB cord and enable USB debugging (from developer options) on the device to use USB debugging. After that, one may use Android Studio to launch and debug your app directly on the device.

If one doesn't want to utilize a USB cable, they may use Android Studio to debug your app wirelessly. To accomplish this, connect your Android device to the same network as your computer and enable wireless debugging under the device's Developer settings. Then, from the Android Studio toolbar, select the device to run and debug your app wirelessly.

If one doesn't have a physical Android device, they may test and debug their app using an emulator. One may use the Android Studio emulator to mimic various Android device setups and test their app on them. Using any of the various ways, debug and run the application on different devices to identify errors or complications in the application.

4.9 Devanagari Character Recognition

Devanagari script is an abugida script used in South Asia to write a variety of languages such as Hindi, Nepali, Marathi, and Sanskrit. It is the official script for the Hindi and Nepali languages and is one of the most extensively used writing systems in the Indian subcontinent. Figure 9 is shown below.
INPUT 1 - printed devanagari script.

INPUT 2 - handwritten devanagari script with very low-quality image.

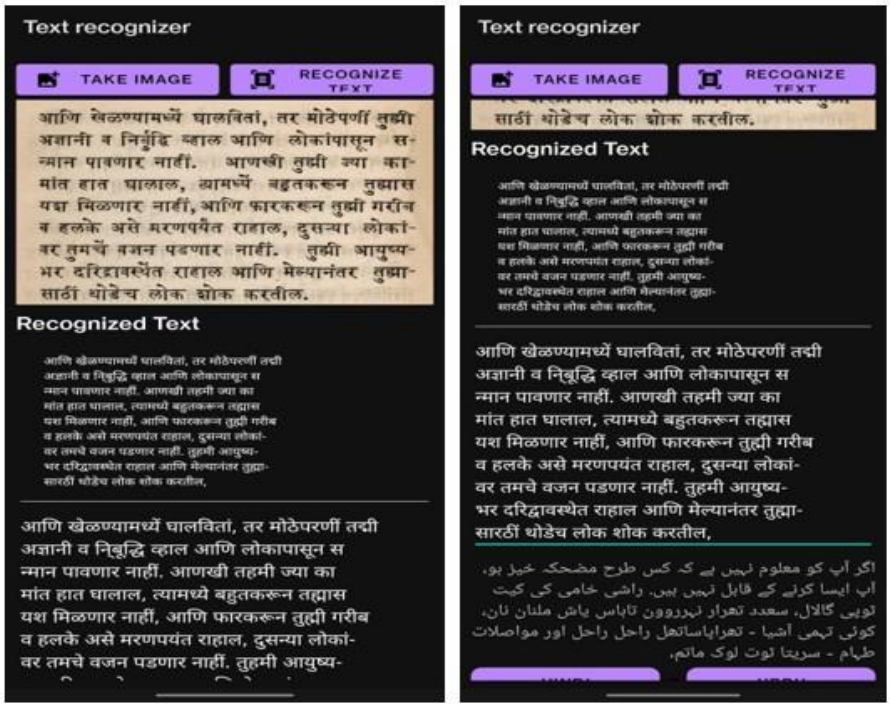

Figure 9: Devanagari Character Recognition and Translation.

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2025, 10(41s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

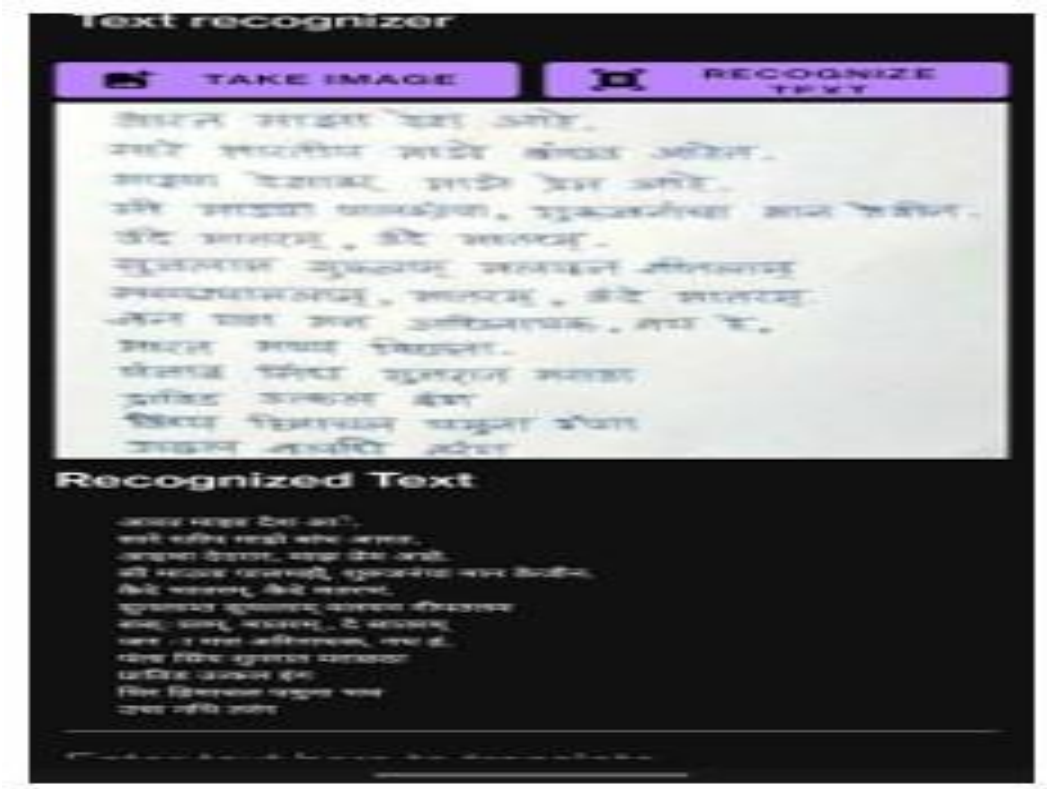

Figure 10: Handwritten Devanagari Character Recognition.

Figure 10 is shown above. Here as seen both printed and handwritten Devanagari script is recognised and translated. This app provides the user to translate to/from 59 languages in offline mode.

4.10 Chinese Character Recognition

By changing just one line of code in the main activity one can convert the recognition to Chinese character recognition (changing the builder). Figure 10 is shown above.

Here by changing the textRecogniser to our desired language, it would detect or recognise characters of that particular language, as shown below in Figure 11 and code.

textRecogniser=TextRecognition.getClient(ChineseTextRecogniserOptions.Builder

(). Build ())

INPUT1 - Printed Chinese paragraph.

INPUT2 - Printed Chinese paragraph for translation into Tamil.

INPUT3 - Handwritten Chinese character recognition and translation to telugu.

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2025, 10(41s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

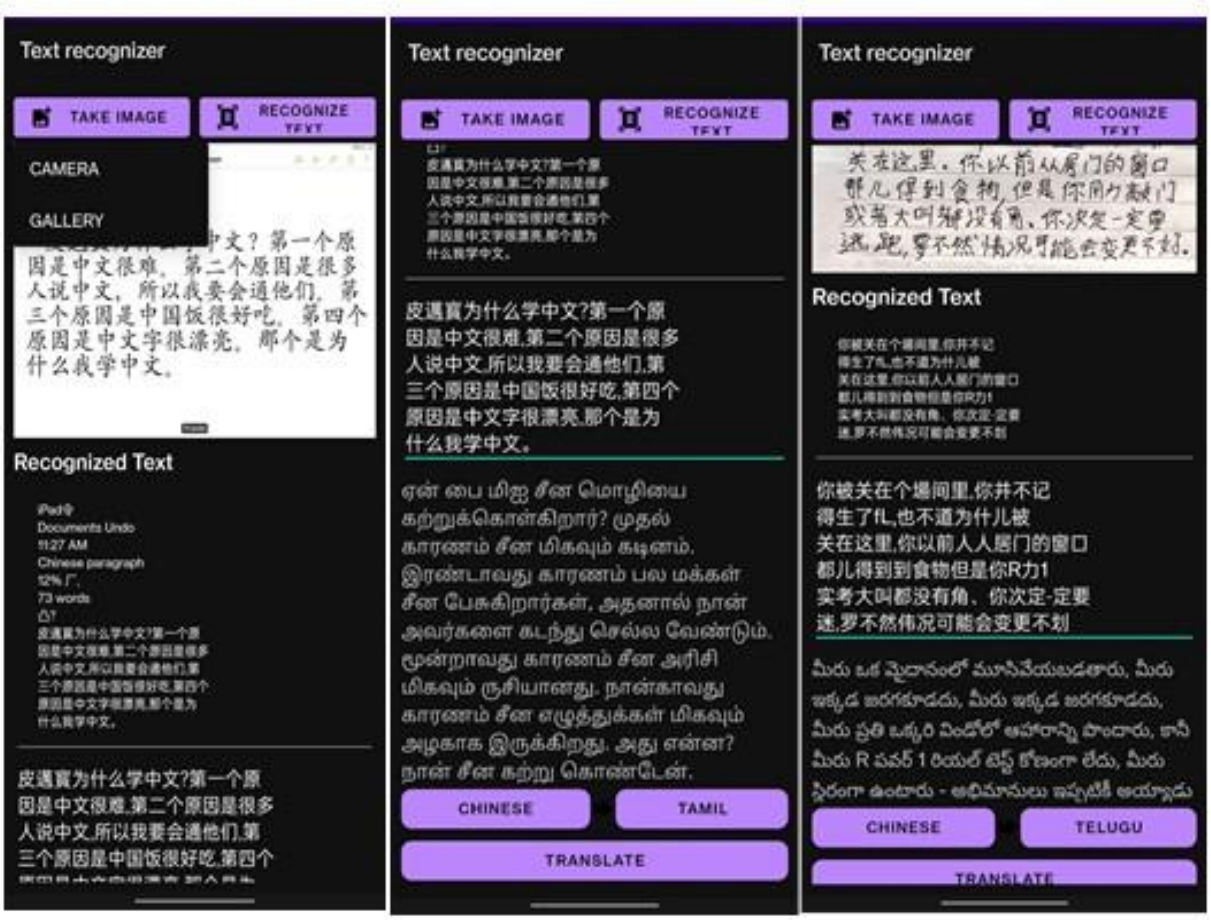

Figure 11: Chinese Character Recognition and Translation.

4.11 Advantages over Google Lens

Users may believe that this software performs the same functions as Google Lens, so why use it instead of Google Lens? This app offers many advantages over Google lens as the app provides a user-friendly interface for recognition and translation where users can quickly capture images, recognise text and translate them without going through additional features as in Google lens. This app can be used in offline mode which is not offered by Google lens. By not requiring Firebase, this app can provide an alternative to Google Lens that is less expensive and more privacy-focused. Data security and privacy issues may arise when Google Lens requests the user's consent to use the camera and microphone. Instead, this app provides users greater control over their privacy by letting them decide when and how the camera and microphone are utilized (least or no chances of data theft). One of the major advantages of this app is the ability to alter the ML model. Programmers may use ML Kit to train their own model on a particular data collection for complex projects.

CONCLUSION

The character recognition app using ML Kit and translator without Firebase implementation in Android Studio, Kotlin and XML languages is demonstrated in this paper. The programme can recognise characters in real time and storage-based recognition then translate them into the required language, making it a handy tool for language learners and anyone who needs to swiftly translate information. The ML Kit is used for character recognition, and the Google Translate API was integrated for translation. The app was created on Android Studio using Kotlin and XML programming languages, and the implementation process included creating the project, adding dependencies, designing the user interface, and manual testing the app for accuracy and functioning in different environments. The objective for the future is to continue increasing the accuracy and efficiency of our character recognition algorithms, as well as to include new features and functions that will make the app even more valuable to users.

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Journal of Information Systems Engineering and Management
2025, 10(41s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

REFRENCES

[1] H. J. Kim et al., "Performance comparison of Google Cloud Translation API and ML Kit Translation API for mobile applications," *International Journal of Advanced Science and Technology*, vol. 29, no. 7, pp. 958-968, 2020.

[2] Sandhya Arora, Debotosh Bhattacharjee, Mita Nasipuri, L. Malik, M. Kundu and D.K. Basu, "Performance Comparison of SVM and ANN for Handwritten Devanagari Character Recognition*", International Journal of Computer Science Issues*, pp 18-26, Vol. 7, Issue 3, No. 6, may 2010.

[3] U.pal and R.B. Chaudhuri, "Indian Script Character Recognition: A Survey", *Pattern recognition*, pp 1887-99, Vol. 37, 2004

[4] Manoj Kumar Shukla, Dr. Haider Banka, "An Efficient Segmentation Scheme for the Recognition of Printed Devanagari Script", *International Journal of Computer Science and Technology*, pp. 529-531, Vol. 2, Issue 4,

[5] Luca Ardito, Riccardo Coppola, Giovanni Malnati, and Marco Torchiano. Effectiveness of kotlin vs. java in android app development tasks. *Information and Software Technology*, 127:106374, 2020

[6] Malanker A. and Patel M. 2014.Handwritten Devanagari Script Recognition: A Survey. *IOSR Journal of Electrical and Electronics Engineering,* (IOSR-JEEE) e-ISSN: 2278- 1676, p-ISSN: 2320-3331 Volume 9, Issue 2 Ver. II, PP 80-87.

[7] Indira B. and Sudha T. 2010. A Pragmatic Approach for Reading Number Plates of Indian Vehicles. *International Journal of Neural Networks and Applications*, 3(1), pp. 15-18.

[8] Sharma, P., 2018. A Reviewm on Devanagari Character Recognition. *IJRAR-International Journal of Research and Analytical Reviews (IJRAR),* 5(3), pp.473-478

[9] Trier D., Jain A.K.and Taxt T. 1996. Feature Extraction Method for Character Recognition – A Survey. Pattern Recognition, pp. 641-662, Vol. 29, No. 4

[10] Bansal and Sharma D. 2010.Isolated Handwritten Words Segmentation Techniques in Gurmukhi Script. *International Journal of Computer Applications*, Vol. 1, No. 24, pp. 104-111.

[11] Kumar M.,Jindal M.K. and Sharma R.K. 2014. Segmentation of Isolated and Touching Characters in Offline Handwritten Gurmukhi Script Recognition. *International Journal Information Technology and Computer Science*, pp.

[12] Garg N.K., Kaur L. and Jindal M.K. 2011. The segmentation of half characters in Handwritten Hindi Text. *Springer Verlag Berlin Heidelberg*, pp. 48-53.

[13] T. Hennig et al., "ML Kit: Enabling Machine Learning on Mobile Devices," *IEEE Pervasive Computing,* vol. 19, no. 2, pp. 26-35, 2020.

[14] M. Benois-Pineau et al., "Real-time object recognition on mobile devices with the Google ML Kit," *Proceedings of the 8th ACM Multimedia Systems Conference*, pp. 452-457, 2017.

[15] N. B. Shridhar et al., "Mobile Machine Learning: Techniques and Applications," *Proceedings of the 2019 IEEE Global Humanitarian Technology Conference*, pp. 1-8, 2019.

[16] Al-Shridah, N. and Sharieh, A., 2002. Recognition process of handwritten and typed Arabic letters. *ADVANCES IN MODELLING AND ANALYSIS-B-*, *45*(1/2), pp.1-1.

Copyright © 2024 by Author/s and Licensed by JISEM. This is an open access article distributed under the Creative Commons Attribution License which permitsunrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.