SIGN LANGUAGE DETECTOR ANDROID APP

Introduction:

The “sign language detector” app aims to provide a means of communication between people who cannot speak and those who cannot understand sign language as it provides speech-to-text functionality as well as real-time sign detection. It uses a machine learning model using the famous frameworks for Android which uses computer vision and natural language processing to convert the speech to text as well as interpreting the hand signs into text, making it easier for the users to communicate with each other through it.

Whether you're a sign language interpreter, a person with hearing impairments, or simply someone interested in communication environments, this app provides a user-friendly interface and robust functionality to facilitate smooth interactions. By incorporating the power of AI and mobile applications, the Hand Sign Language Detector offers a practical solution for breaking down communication barriers and promoting greater understanding and empathy in diverse social contexts.

This documentation contains the working of the Sign Language Detector app and, the frameworks it incorporates as well as a guide to setting up the project on Android Studio.

Technicality:

The working and composition of this app are broken down into various components. We will start by understanding the building blocks and then finally get to the working of the entire application.

TensorFlow Lite: TensorFlow Lite is a lightweight machine learning framework that allows developers to deploy machine learning models into end devices such as mobile apps, web apps, or embedded systems by converting ML/ DL models into TensorFlow Lite format. TensorFlow Lite enables efficient inference of machine learning models, making it ideal for real-time applications such as object detection, image recognition, natural language processing, and gesture detection. In the context of the application that we have developed, this framework plays an important role in incorporating our trained hand sign detection model into the Android app using the MediaPipe library for making inferences about the hand signs given as input.

MediaPipe: MediaPipe for Android is a powerful framework developed by Google that enables developers to build real-time multimedia processing applications on Android devices. It provides a comprehensive set of tools and pre-built components for tasks such as image processing, object detection, hand tracking, pose estimation, and more. It is used to incorporate pre-trained ML / DL models into the Android app pipeline.

Combining it with TensorFlow Lite, we can achieve high-performance hand sign detection and interpretation directly on our mobile device, ensuring optimal user experience.

CameraX API: It is obvious that to use the deployed model for hand sign detection on our Android app, we would need some way to pass the frames or images as input to the model in real time. This is why we have incorporated the use of CameraX API for Android.

CameraX API is a Jetpack library provided by Google, designed to simplify the process of integrating camera functionalities into Android applications. Particularly, we have used the "Image Analysis" function of this library to render the camera frames as bitmaps to be passed to the model to make predictions and to show the rendered frames on the screen. As a result, by using this library, we can focus on the functionalities of our Android application while benefiting from robust and reliable camera functionality provided by Google's Jetpack library.

Now that we have understood the basic building blocks of this app, let's take a look at how this app is basically structured. The following points explain the process of building this application:

- **Training the model:** As the first stage of training any model, the data is collected. In this case, various images for different hand signs are collected in such a way as to diversify the dataset as much as possible to make predictions as accurate as possible. After the data collection, the mediapipe model maker is used to train the model and fine-tune it using hyperparameter tuning and quantization to make it more efficient.
  The model that is trained is the "MobileNet-MultiHW-AVG" model which is a Convolutional Neural Network model used for vision tasks. It is built upon previous MobileNet architecture models, but this model provides efficiency in making inferences and is better in terms of latency and accuracy as compared to the previously developed models of the same family.
  Finally, after training, the model in TensorFlow Lite format is obtained for use in our Android app.
- **Configuring CameraX API:** Next, we have to set up our camera that takes input in the form of a bitmap. For that, we have configured the CameraX in our project to use both the front and back cameras of our device. Furthermore, The "Image Analyzer" is used to fetch the bitmap of the camera output, to feed it to our model to make a prediction, and then finally to show it as an output on the screen.
- **Using the MediaPipe library:** Now we need to set up the object detector use-case from the mediapipe library which will use our trained model to detect the hand signs. For that, we have set it up for image inputs for which we will use the extracted bitmap using the camerax's "Image Analyzer". For creating the instance of the mediapipe's object detector class, we will need to set some options like how many predictions we want to make at a time, whether we would like to use CPU or GPU for making detections and the threshold for predictions and so on

(Mediapipe android documentation can be read for details). For this case, we will be making one prediction at a time while using CPU as delegate with threshold value of around 20-30%. After setting up these options, we will instantiate the object detector class for use. After that, we will pass the bitmap input to the object detector's detect method which returns the predictions as a list. From this list, we will extract the predicted class (or hand sign in our case) and display it to a text field on the screen. This field will also be updated in real-time.

Feature Overview:

The following are the main features of this app:

- **Realtime hand sign detection:** Detects the hand signs of the user and displays the prediction in real-time.
- **Speech-to-text support:** Allowing users to convert spoken language into text. This feature enables seamless communication between individuals who use sign language and those who rely on spoken language, bridging communication barriers.
- **User-Friendly Interface:** Provides a user-friendly interface. Making it easier for the user to navigate through the app easily and quickly.

User-guide:

In order to import the project in android studio, following steps are to be followed:

- **Install Android Studio:** If you haven't already done so, download and install Android Studio from the official website (https://developer.android.com/studio). Follow the installation instructions provided for your operating system.
- **Open Android Studio:** Once Android Studio is installed, launch the application by locating it in your applications folder (on macOS) or through the Start menu (on Windows). You can also search for "Android Studio" in your computer's search bar.
- **Set Up Android Studio:** Upon opening Android Studio for the first time, you may be prompted to import settings or customize the setup. Follow the on-screen instructions to configure Android Studio according to your preferences.
- **Import Project:** If you already have an existing Android Studio project that you want to open, select "Open an existing Android Studio project" from the welcome screen. Navigate to the directory where your project is located and select the project folder. Click "OK" to import the project into Android Studio.
- **Sync Project with Gradle Files:** Once your Hand Sign Detection app project is imported, Android Studio will automatically sync the project with its Gradle build files. This process may take a few moments as Android Studio downloads any necessary dependencies and sets up the project's build configuration.

- **Build and Run Project:** After the project is synced, you can build and run your Android Studio project by clicking the green "Run" button (with a play icon) in the toolbar. Android Studio will compile your project, build the APK (Android Package) file, and deploy it to the selected emulator or connected Android device for testing.
- **Explore Project Structure:** Take some time to familiarize yourself with the project structure and files in Android Studio. The project navigator on the left side of the screen allows you to navigate through the various components of your project, including source code, resources, and configuration files.

References:

- https://developers.google.com/mediapipe/framework/getting_started/android
- https://developers.google.com/mediapipe/solutions/vision/object_detector/android
- https://www.tensorflow.org/lite/android
- https://developer.android.com/studio/intro
- https://developer.android.com/media/camera/camerax
- https://console.cloud.google.com/vertex-ai/publishers/google/model-garden/100?project=gen-lang-client-0417801515