LabW07 – Android ML-kit

Objectives:

1. Get familiar with the ML Kit provided by Google.
2. Familiar with implementing simple ML-based operations on Android to solve real-world problems.

Tasks:

1. Add ML Kit SDK to your Android device.
2. Create simple language translation and image labelling application.

.
ML Kit is a ready to use SDK provided by Google which contains different ML APIs for both vision and language processing-based tasks such as text recognition, face detection, image labelling, text translation, etc. ML Kit's APIs all run on-device, allowing for real-time use cases where you want to process a live camera stream, for example. This also means that the functionality is available offline.

Task 1: Natural language processing: Translate text from one language to another.

In this task, we will implement a simple text translation application using ML Kit on device Translational API that comes under natural language processing and . This API provides you translations over 50 languages. In the following application, a given input text in English language will be translated to four different languages upon your selection.

1. On the '**No Activity'** Screen Enter the following
    a. **Name**: MLTranslator
    b. **Package name**: comp5216.sydney.edu.au.mltranslator
    c. **Language:** JAVA
    d. **Minimum SDK:** API 24: Android 7.0 (Nougat)
    e. **Build configuration language:** Groovy DSL

2. In your project-level build.gradle include the Google's Maven repository in buildscript section.

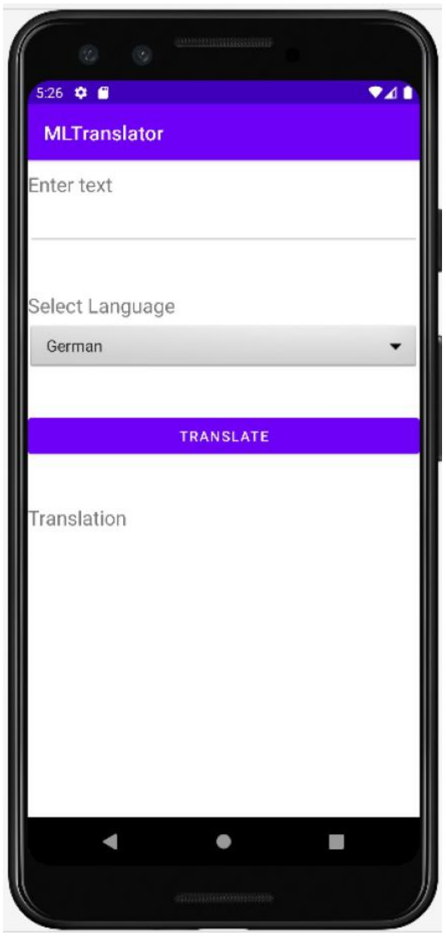
```
buildscript {
    repositories {
        google()
        mavenCentral()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:7.4.2'
    }
}
```

3. Add the dependencies for the ML Kit Android libraries to your module's app-level gradle file, which is usually **app/build.gradle**. This dependency includes ML kit libraries for translation API to your project.

```
dependencies {

...

    implementation 'com.google.mlkit:translate:17.0.1'

}
```

4. Create the following simple layout which mainly contains **EditText** to input the source text, **Spinner** for drop-down list to select different languages, and **TextView** to contain the output translated text. Sample **activity_main.xml** is given in Appendix. (Note: Dropdown list or **Spinner** will be populated by different language names later in **MainActivity.java**)

5. Run the code. You will see a following UI.



6. In your MainActivity.java file, inlcude following member variables to hold the text strings and to connect to the UI elements.

```
// variables to connect with user elements
Spinner mSpinnerLanguages;
// text box to input text in English
EditText mInputText;
// textview to show the translated text
TextView mTranslatedText;

// variable holding the langauge name that the input string should be translated
to
String mTransLang;
// variable to hold the input text string in English
String mInputString;
```

7. We will translate input text in English language to German, French, Chinese and Hindi languages. Therefore to populate the drop-down list (**Spinner)** first, include following strings variable in **strings.xml** file as a **string-array.**

```
<resources>
    <string name="app_name">MLTranslator</string>
    <string-array name="languages">
        <item>German</item>
        <item>French</item>
        <item>Chineese</item>
        <item>Hindi</item>

    </string-array>
</resources>
```

Populate the **Spinner** adding the above language strings as follows in **onCreate()** function. **mTransLanag** which holds the language that input text to be translated

```
// Link the input text and the output translation text view
mInputText = (EditText) findViewById(R.id.input_text);
mTranslatedText = (TextView) findViewById(R.id.tranaslation);

// Populate the dropdown list
mSpinnerLanguages = findViewById(R.id.spinner1);
ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(this,
R.array.languages, android.R.layout.simple_spinner_item);
adapter.setDropDownViewResource(android.R.layout.simple_spinner_item);
mSpinnerLanguages.setAdapter(adapter);
// get the default language as the German
mTransLang = mSpinnerLanguages.getSelectedItem().toString();
```

8. Next, add the following listner function to track the changes in the selection of the languages in the drop-down list. For this purpose we use **setOnItemSelectedListner(…)** from Spinner class.

```
public void startSpinnerListner() {
    mSpinnerLanguages.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
        @Override
        public void onItemSelected(AdapterView<?> parentView, View
selectedItemView, int position, long id) {
            // your code here
            // assign the currently seelcted language to the member variable
            mTransLang = mSpinnerLanguages.getSelectedItem().toString();
        }

        @Override
        public void onNothingSelected(AdapterView<?> parentView) {
            // your code here
        }
    });
}
```

9. To start translating the input string to the selected language, add the following function, **translateText(View view)** to **MainActivity.java** class.

   Step 1: In this function first, we need to set target language which is returned by a helper function called **getTranslationLang(String lang)** given below.

   Step 2: You will create a Translator object, configuring it with the source and target languages.

   Step 3: You have to make sure that required translation model has been downloaded to the device. Note that Language models are around 30MB, so don't download them unnecessarily, and only download them using Wi-Fi unless the user has specified otherwise.

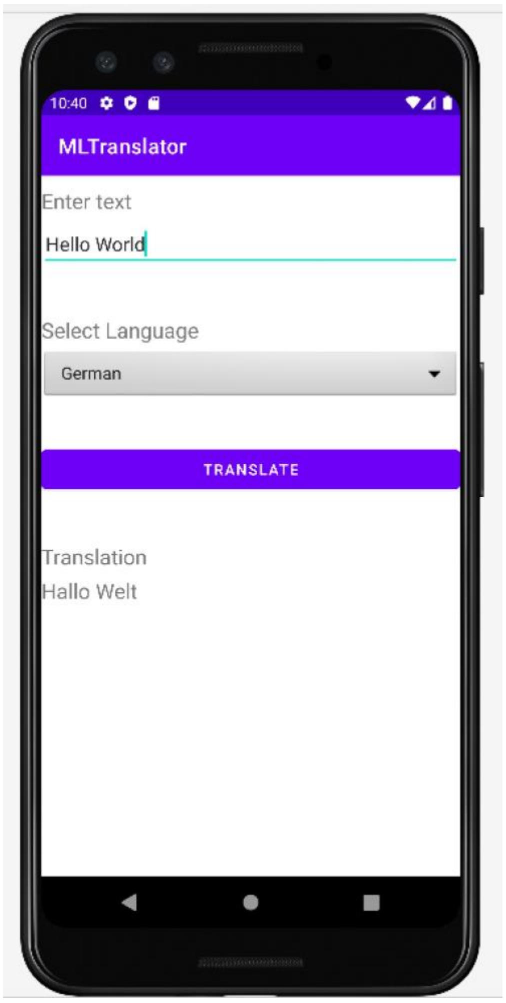
   Step 4: After you confirm the model has been downloaded, execute **runTranslation()** method which passes the text in source language to **translate().**

```
public void translateText(View view) {

    //Step 1
    // get the Target language according to the drop down list selection.
    String constTranslateLang;
    constTranslateLang = getTranaslationLang(this.mTransLang);

    // Step 2
    // Create a Translator object, configuring it with the source and target
languages
    TranslatorOptions options = new TranslatorOptions.Builder()
            .setSourceLanguage(TranslateLanguage.ENGLISH)
            .setTargetLanguage(constTranslateLang)
            .build();
    englishOtherTranslator = Translation.getClient(options);

    // Step 3
    // Make sure the required translation model has been downloaded to the
device
    DownloadConditions conditions = new DownloadConditions.Builder()
            .requireWifi()
            .build();
    englishOtherTranslator.downloadModelIfNeeded(conditions)
            .addOnSuccessListener(
                    new OnSuccessListener() {
                        @Override
                        public void onSuccess(Object o) {
                            // Model downloaded successfully. Okay to start
translating.
                            //Step 4:
                            runTranslation();

                            Toast
toast=Toast.makeText(getApplicationContext(),"Model downloaded
successfully",Toast.LENGTH_LONG);
                            toast.setMargin(50,50);
                            toast.show();
                        }
                    })
            .addOnFailureListener(
                    new OnFailureListener() {
                        @Override
                        public void onFailure(@NonNull Exception e) {
                            // Model couldn't be downloaded or other internal
error.
                            Toast toast =
Toast.makeText(getApplicationContext(), "Model downloaded Failed",
Toast.LENGTH_LONG);
                            toast.setMargin(50, 50);
                            toast.show();

                        }
                    });
}
```

```
public String getTranaslationLang(String lang){
    String constTranslateLang = null;

    switch (lang) {
        case "Hindi":
            constTranslateLang = TranslateLanguage.HINDI;
            break;
        case "French":
            constTranslateLang = TranslateLanguage.FRENCH;
            break;
        case "Chineese":
            constTranslateLang = TranslateLanguage.CHINESE;
            break;
        default:
            constTranslateLang = TranslateLanguage.GERMAN;
    }
    return constTranslateLang;
}
```

```
public void runTranslation(){
    mInputString = mInputText.getText().toString();
    Task<String> result = englishOtherTranslator.translate(mInputString)
            .addOnSuccessListener(
                    new OnSuccessListener<String>() {
                        @Override
                        public void onSuccess(String translatedText) {

                            mTranslatedText.setText(translatedText);

                            // Translation successful.
                            Toast toast =
Toast.makeText(getApplicationContext(), "Translation successful",
Toast.LENGTH_LONG);
                            toast.setMargin(50, 50);
                            toast.show();

                        }
                    })
            .addOnFailureListener(
                    new OnFailureListener() {
                        @Override
                        public void onFailure(@NonNull Exception e) {
                            // Translation successful.
                            Toast toast =
Toast.makeText(getApplicationContext(), "Translation unsuccessful",
Toast.LENGTH_LONG);
                            toast.setMargin(50, 50);
                            toast.show();
                        }
                    });
}
```

10. Run your application. You can change different languages to translate and see the output.



Task 2: Image processing: Identify labels in an image

In this task we leverage ML Kit's image labeling APIs which can be used to detect and extract information about entities in an image across a broad group of categories. The default image labeling model can identify general objects, places, activities, animal species, products, and more.

In the follow, you will manually add an image to your app folder and extract the image labels on it.

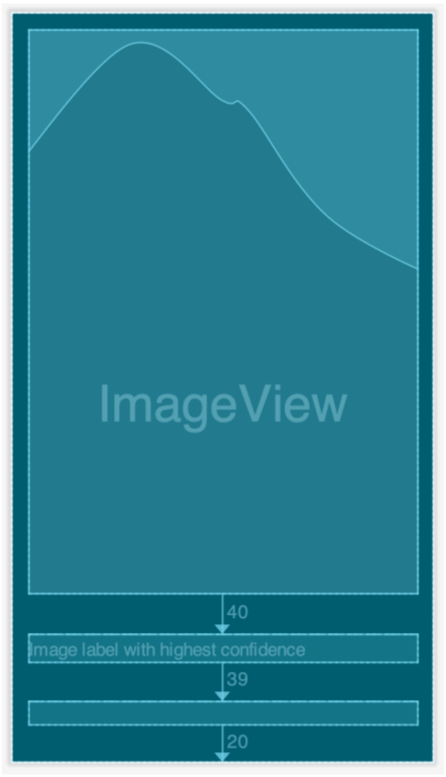
1. On the 'NO Activity' Screen Enter the following
    a. **Name:** ImageLabler
    b. **Package name:** comp5216.sydney.edu.au. imagelabler
    c. **Language:** JAVA
    d. **Minimum SDK:** API 24: Android 7.0 (Nougat)
    e. **Build configuration language:** Groovy DSL

2. In your project-level **build.gradle** include the Google's Maven repository in buildscript section. (Same as Task 1- step 2)

3. Add the dependencies for the ML Kit Android libraries to your module's app-level gradle file, which is usually **app/build.gradle**. Choose one of the following dependencies based on your needs

```
dependencies {

    implementation 'com.google.mlkit:image-labeling:17.0.7'
}
```

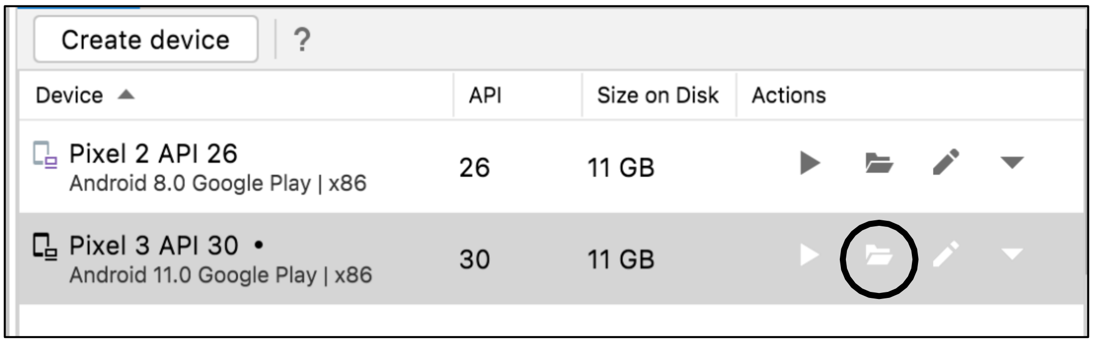
4. Use the **activity_main.xml** given in Appendix for Task 2 to create the following UI which contains the image view, and text view to display the image label with highest confidence level.



5. Run the code to see the corresponding UI for the above layout.
6. Include following variable for the **MainActivity.java** class and initialize them in the **oncreate()** method.

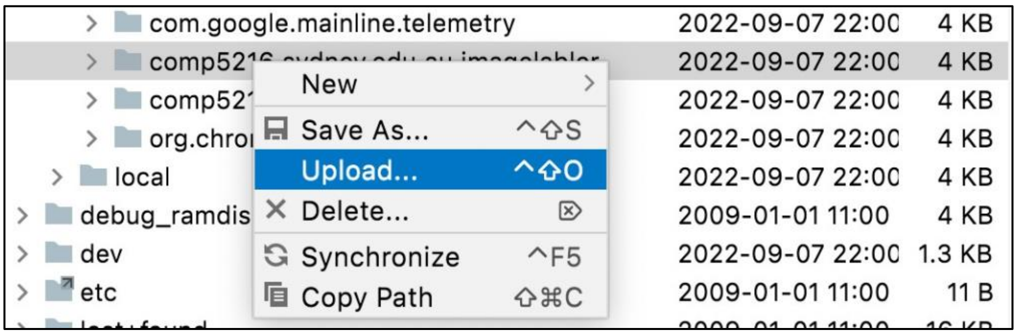
```
// variable to hold the image and the directory
InputImage mImageToProcess;
File mediaStorageDir;

// variable to hold the image view and the high confidence label view
ImageView ivPreview;
TextView mHighConfidence;

protected void onCreate(Bundle savedInstanceState) {
    ...
    ivPreview = (ImageView) findViewById(R.id.photopreview);
    mHighConfidence = (TextView) findViewById(R.id.labelview_confidence_output);

}
```

7. To include the sample images, open the **Device File Explorer** in your running emulator using either **Device Manger** panel or **View➔Tool Windows➔Device File Explorer**



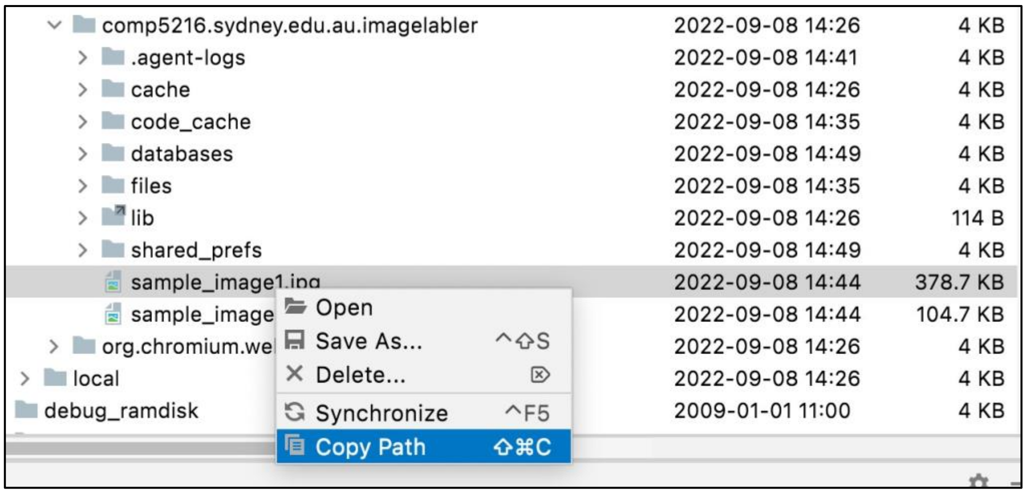
Device	API	Size on Disk	Actions
Pixel 2 API 26 Android 8.0 Google Play | x86	26	11 GB	
Pixel 3 API 30 • Android 11.0 Google Play | x86	30	11 GB	

Go inside to the **data→data→ comp5216.sydney.edu.au.imagelabler** Right click on comp5216.sydney.edu.au.imagelabler folder and upload the sample images we have given with the lab material. (sample_image1.jpg and sample_image2.jpg)


com.google.mainline.telemetry 2022-09-07 22:00 4 KB
comp5216.sydney.edu.au.imagelabler 2022-09-07 22:00 4 KB
comp52 2022-09-07 22:00 4 KB
org.chro 2022-09-07 22:00 4 KB
local 2022-09-07 22:00 4 KB
debug_ramdis 2009-01-01 11:00 4 KB
dev 2022-09-07 22:00 1.3 KB
etc 2009-01-01 11:00 11 B
New
Save As... ^⇧S
Upload... ^⇧O
Delete...
Synchronize ^F5
Copy Path ⇧⌘C

8. To read the sample image from the device include **get_image()** method given below. Since we just focus on running image labelling API on an existing image, copy the absolute path of the image as shown in function. You can right click on the image in file directory to copy the path.

```
public InputImage get_image(){

    String image_path =
"/data/data/comp5216.sydney.edu.au.imagelabler/sample_image1.jpg";

    mediaStorageDir  = new File(image_path);
    Uri fileUri = Uri.fromFile(mediaStorageDir);
    InputImage image = null;
    try {
        image = InputImage.fromFilePath(this.getApplicationContext(), fileUri);
    } catch (IOException e) {
        e.printStackTrace();
    }

    return image;
}
```



```
comp5216.sydney.edu.au.imagelabler   2022-09-08 14:26   4 KB
  .agent-logs                        2022-09-08 14:41   4 KB
  cache                              2022-09-08 14:26   4 KB
  code_cache                         2022-09-08 14:35   4 KB
  databases                          2022-09-08 14:49   4 KB
  files                              2022-09-08 14:35   4 KB
  lib                                2022-09-08 14:26   114 B
  shared_prefs                       2022-09-08 14:49   4 KB
  sample_image1.jpg                  2022-09-08 14:44   378.7 KB
  sample_image                       2022-09-08 14:44   104.7 KB
org.chromium.we                      2022-09-08 14:26   4 KB
local                                2022-09-08 14:26   4 KB
debug_ramdisk                        2009-01-01 11:00   4 KB
```

9. Add the following method which creates **ImageLabeling** client to run the image labelling process. This also preview the image you have selected and image label with the highest confidence level. See the logcat with “TAG” search query to see the other labels provided.

```
public void process_image(InputImage image){
    ImageLabeler labeler =
ImageLabeling.getClient(ImageLabelerOptions.DEFAULT_OPTIONS);

    labeler.process(image)
            .addOnSuccessListener(new OnSuccessListener<List<ImageLabel>>() {
                @Override
                public void onSuccess(List<ImageLabel> labels) {
                    // Task completed successfully
                    // ...
                    for (ImageLabel label : labels) {
                        String text = label.getText();
                        Log.d("TAG",text);
                    }

                    String firstLabel = String.valueOf(labels.get(0).getText());
                    String firstConfidence =
String.valueOf(labels.get(0).getConfidence());
                    String highConfidentLabel =" Label: " + firstLabel + "
Conf: "+ firstConfidence;
                    mHighConfidence.setText(highConfidentLabel);

                    Bitmap takenImage =
BitmapFactory.decodeFile(mediaStorageDir.getAbsolutePath());
                    // Load the taken image into a preview
                    ivPreview.setImageBitmap(takenImage);
                    ivPreview.setVisibility(View.VISIBLE);

                }
            })
            .addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception e) {
                    // Task failed with an exception
                    // ...
                }
            });
}
```

10. Include **get_image()** and **process_image()** methods in the **onCreate()** method.

```
    protected void onCreate(Bundle savedInstanceState) {

        ...
        mImageToProcess = get_image();
        process_image( mImageToProcess);
    }
```

11. Run the code. You will be able to see an output as below.

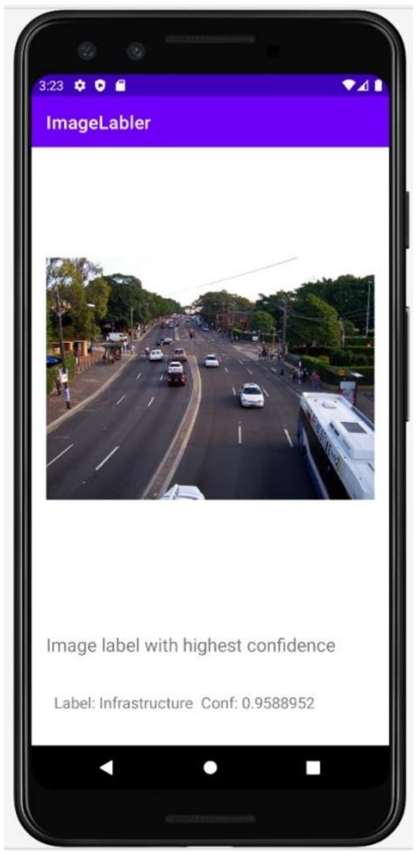


Image label with highest confidence

Label: Infrastructure Conf: 0.9588952

References

- ML toolkit that contains all the related coding materials for Vision and Natural Language processing based ML tasks.

  https://developers.google.com/ml-kit/guides

Appendix

Task 1: **activity_main.xml**

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">


    <TextView
        android:id="@+id/label_input_text"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:layout_marginTop="10dp"
        android:text= "Enter text"
        android:textSize="20dp" />


    <EditText
        android:id="@+id/input_text"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/label_input_text"
        android:layout_marginTop="5dp"
        android:text="" />

    <TextView
        android:id="@+id/label_sel_language"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/input_text"
        android:layout_marginTop="44dp"
        android:text = "Select Language"
        android:textSize="20dp" />


    <Spinner
        android:id="@+id/spinner1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/label_sel_language"
        android:layout_marginTop="5dp"
        android:background="@android:drawable/btn_dropdown"
        android:spinnerMode="dropdown" />
    <Button
        android:id="@+id/translate"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/spinner1"
        android:layout_marginTop="40dp"
        android:text="Translate" />


    <TextView
        android:id="@+id/label_translation"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/translate"
        android:layout_marginTop="44dp"
        android:text = "Translation"
        android:textSize="20dp" />

    <TextView
        android:id="@+id/tranaslation"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@+id/label_translation"
        android:layout_marginTop="5dp"
        android:textSize="20dp" />
</RelativeLayout>
```

Task 1: MainActivity.java

```
package comp5216.sydney.edu.au.mltranslator;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

//import android.util.Log;
import android.text.Editable;
import android.text.TextWatcher;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.EditText;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.mlkit.common.model.DownloadConditions;
import com.google.mlkit.common.model.RemoteModelManager;
import com.google.mlkit.nl.translate.TranslateLanguage;
import com.google.mlkit.nl.translate.TranslateRemoteModel;
import com.google.mlkit.nl.translate.Translation;
import  com.google.mlkit.nl.translate.Translator;
import com.google.mlkit.nl.translate.TranslatorOptions;

import java.util.Set;

public class MainActivity extends AppCompatActivity {

    // variables to connect with user elements
    Spinner mSpinnerLanguages;
    // text box to input text in English
    EditText mInputText;
    // textview to show the translated text
    TextView mTranslatedText;

    // variable holding the langauge name that the input string should be
translated to
    String mTransLang;
    // variable to hold the input text string in English
    String mInputString;

    Translator englishOtherTranslator;

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

        // Link the input text and the output translation text view
        mInputText = (EditText) findViewById(R.id.input_text);
        mTranslatedText = (TextView) findViewById(R.id.tranaslation);

        // Populate the dropdown list
        mSpinnerLanguages = findViewById(R.id.spinner1);
        ArrayAdapter<CharSequence> adapter =
ArrayAdapter.createFromResource(this, R.array.languages,
android.R.layout.simple_spinner_item);
        adapter.setDropDownViewResource(android.R.layout.simple_spinner_item);
        mSpinnerLanguages.setAdapter(adapter);
        // get the default language as the German
        mTransLang = mSpinnerLanguages.getSelectedItem().toString();

        // start listner function for the dropdown list (spinner) and input text
        startSpinnerListner();

    }

    public void startSpinnerListner() {
        mSpinnerLanguages.setOnItemSelectedListener(new
AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parentView, View
selectedItemView, int position, long id) {
                // your code here
                // assign the currently seelcted language to the member variable
                mTransLang = mSpinnerLanguages.getSelectedItem().toString();
            }
            @Override
            public void onNothingSelected(AdapterView<?> parentView) {
                // your code here
            }
        });
    }

    public String getTranaslationLang(String lang){
        String constTranslateLang = null;

        switch (lang) {
            case "Hindi":
                constTranslateLang = TranslateLanguage.HINDI;
                break;
            case "French":
                constTranslateLang = TranslateLanguage.FRENCH;
                break;
            case "Chineese":
                constTranslateLang = TranslateLanguage.CHINESE;
                break;
            default:
                constTranslateLang = TranslateLanguage.GERMAN;
        }
        return constTranslateLang;
    }

    public void translateText(View view) {

        //Step 1
        // get the Target language according to the drop down list selection.
        String constTranslateLang;
        constTranslateLang = getTranaslationLang(this.mTransLang);

        // Step 2
        // Create a Translator object, configuring it with the source and target
languages
        TranslatorOptions options = new TranslatorOptions.Builder()
                .setSourceLanguage(TranslateLanguage.ENGLISH)
                .setTargetLanguage(constTranslateLang)
                .build();
        englishOtherTranslator = Translation.getClient(options);

        // Step 3
        // Make sure the required translation model has been downloaded to the
device
        DownloadConditions conditions = new DownloadConditions.Builder()
                .requireWifi()
                .build();
        englishOtherTranslator.downloadModelIfNeeded(conditions)
                .addOnSuccessListener(
                        new OnSuccessListener() {
                            @Override
                            public void onSuccess(Object o) {
                                // Model downloaded successfully. Okay to start
translating.
                                //Step 4:
                                runTranslation();
```

```
                                    Toast
toast=Toast.makeText(getApplicationContext(),"Model downloaded
successfully",Toast.LENGTH_LONG);
                                    toast.setMargin(50,50);
                                    toast.show();
                                }
                            })
                    .addOnFailureListener(
                            new OnFailureListener() {
                                @Override
                                public void onFailure(@NonNull Exception e) {
                                    // Model couldn't be downloaded or other
internal error.
                                    Toast toast =
Toast.makeText(getApplicationContext(), "Model downloaded Failed",
Toast.LENGTH_LONG);
                                    toast.setMargin(50, 50);
                                    toast.show();

                                }
                            });

    }

    public void runTranslation(){
        mInputString = mInputText.getText().toString();
        Task<String> result = englishOtherTranslator.translate(mInputString)
                .addOnSuccessListener(
                        new OnSuccessListener<String>() {
                            @Override
                            public void onSuccess(String translatedText) {

                                mTranslatedText.setText(translatedText);

                                // Translation successful.
                                Toast toast =
Toast.makeText(getApplicationContext(), "Translation successful",
Toast.LENGTH_LONG);
                                toast.setMargin(50, 50);
                                toast.show();

                            }
                        })
                .addOnFailureListener(
                        new OnFailureListener() {
                            @Override
                            public void onFailure(@NonNull Exception e) {
                                // Translation successful.
                                Toast toast =
Toast.makeText(getApplicationContext(), "Translation unsuccessful",
Toast.LENGTH_LONG);
                                toast.setMargin(50, 50);
                                toast.show();
                            }
                        });
    }
}
```

Task 2: activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingLeft="16dp"
    android:paddingRight="16dp"
    android:paddingTop="16dp"
    android:paddingBottom="16dp"
    tools:context=".MainActivity">


    <ImageView
        android:id="@+id/photopreview"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_above="@+id/labelview"
        android:layout_marginBottom="40dp"
        android:scaleType="fitCenter" />

    <TextView
        android:id="@+id/labelview"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_above="@+id/labelview_confidence_output"
        android:layout_marginBottom="39dp"
        android:textSize="20dp"
        android:text="Image label with highest confidence" />

    <TextView
        android:id="@+id/labelview_confidence_output"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textSize="17dp"
        android:layout_alignParentBottom="true"
        android:layout_marginBottom="20dp"
        />


</RelativeLayout>
```

```
package comp5216.sydney.edu.au.imagelabler;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import  android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.net.Uri;
import android.os.Bundle;
import android.os.Environment;
import android.util.Log;
import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;

import com.google.android.gms.tasks.OnFailureListener;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.android.gms.tasks.Task;
import com.google.mlkit.vision.common.InputImage;
import com.google.mlkit.vision.label.ImageLabel;
import com.google.mlkit.vision.label.ImageLabeler;
import com.google.mlkit.vision.label.ImageLabeling;
import com.google.mlkit.vision.label.defaults.ImageLabelerOptions;

import  java.io.File;
import java.io.IOException;
import java.util.List;

public class MainActivity extends AppCompatActivity {

    // variable to hold the image and the directory
    InputImage mImageToProcess;
    File mediaStorageDir;

    // variable to hold the image view and the high confidence label view
    ImageView ivPreview;
    TextView mHighConfidence;

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

        ivPreview = (ImageView) findViewById(R.id.photopreview);
        mHighConfidence = (TextView)
findViewById(R.id.labelview_confidence_output);

        mImageToProcess = get_image();
        process_image( mImageToProcess);
    }

    public InputImage get_image(){

        String image_path =
"/data/data/comp5216.sydney.edu.au.imagelabler/sample_image1.jpg";
```

```
        mediaStorageDir  = new File(image_path);
        Uri fileUri = Uri.fromFile(mediaStorageDir);
        InputImage image = null;
        try {
            image = InputImage.fromFilePath(this.getApplicationContext(),
fileUri);
        } catch (IOException e) {
            e.printStackTrace();
        }

        return image;
    }

    public void process_image(InputImage image){
        ImageLabeler labeler =
ImageLabeling.getClient(ImageLabelerOptions.DEFAULT_OPTIONS);

        labeler.process(image)
                .addOnSuccessListener(new OnSuccessListener<List<ImageLabel>>()
{
                    @Override
                    public void onSuccess(List<ImageLabel> labels) {
                        // Task completed successfully
                        // ...
                        for (ImageLabel label : labels) {
                            String text = label.getText();
                            Log.d("TAG",text);
                        }

                        String firstLabel =
String.valueOf(labels.get(0).getText());
                        String firstConfidence =
String.valueOf(labels.get(0).getConfidence());
                        String highConfidentLabel ="  Label: " + firstLabel + "
Conf: "+ firstConfidence;
                        mHighConfidence.setText(highConfidentLabel);

                        Bitmap takenImage =
BitmapFactory.decodeFile(mediaStorageDir.getAbsolutePath());
                        // Load the taken image into a preview
                        ivPreview.setImageBitmap(takenImage);
                        ivPreview.setVisibility(View.VISIBLE);

                    }
                })
                .addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {
                        // Task failed with an exception
                        // ...
                    }
                });
    }


}
```