Android Development with Kotlin

Machine Learning

Machine Learning

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it learn for themselves.

The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly.

Machine Learning capabilities and Algorithms can also be added to your android applications to improve upon or add additional features to your android apps.

Firebase ML Kit

Firebase ML Kit is a tool that allows us to add Machine Learning capabilities within our Android Applications. The Ml-Kit offered by Firebase is equipped with a number of ML models and also allows users to create custom models which may better fir their use case. Some of the services offered by the ML Kit are; Text Recognition, Face Detection, Image Labelling, Object Detection etc. All of these models can be run on the device without the use of internet.

Firebase also offers a few of its ML services via a cloud platform. This ensures a higher accuracy, faster prediction and a larger result set. These services include, Text Recognition, Image Labelling and Landmark Identification.

Any developer can use the Firebase ML Kit as a background in ML is not required for pre-built models. You will however require a knowledge of ML to create custom Models for your application.

Android Neural Networks API

Neural Networks are an Artificial Intelligence model that are used to perform Deep Learning operations. The Android Neural Network API allows the developers to create and run custom Neural Networks within their Android Apps. These models are coded in the C language to ensure a faster execution time.

A knowledge of both AI as well as Android Framework is required to use this library.

Text Recognition

The text Recognition API by Firebase ML – Kit allows you to read text from an image. The following steps can be followed to add Text Recognition to your application:

1. In your project-level build.gradle file, make sure to include Google's Maven repository in both your buildscript and allprojects sections.

2. Add the dependencies for the ML Kit Android libraries to your module's app-level gradle file, which is usually app/build.gradle:

```
dependencies {
  // ...
    implementation  'com.google.android.gms:play-services-mlkit-text-recognition:16.0.0'
}
```

3. **Optional but recommended:** You can configure your app to automatically download the ML model to the device after your app is installed from the Play Store. To do so, add the following declaration to your app's AndroidManifest.xml file:

```
<application ...>
  ...
  <meta-data
      android:name="com.google.mlkit.vision.DEPENDENCIES"
      android:value="ocr" />
  <!-- To use multiple models: android:value="ocr,model2,model3" -->
</application>
```

4. Create input image from the bitmap file (can also use other methods as mentioned on the Google's Developer website: https://developers.google.com/ml-kit/vision/text-recognition/android#kotlin)

```
val image = InputImage.fromBitmap(bitmap, 0)
```

5. Get an instance of the TextRecogniser and process the image

```
val recognizer = TextRecognition.getClient()
val result = recognizer.process(image)
        .addOnSuccessListener { visionText ->
            // Task completed successfully
            // ...
        }
        .addOnFailureListener { e ->
            // Task failed with an exception
            // ...
        }
```

6. Extract text from blocks of recognised text

```
val resultText = result.text
for (block in result.textBlocks) {
    val blockText = block.text
    val blockCornerPoints = block.cornerPoints
    val blockFrame = block.boundingBox
    for (line in block.lines) {
        val lineText = line.text
        val lineCornerPoints = line.cornerPoints
        val lineFrame = line.boundingBox
        for (element in line.elements) {
            val elementText = element.text
            val elementCornerPoints = element.cornerPoints
            val elementFrame = element.boundingBox
        }
    }
}
```

To use a cloud based recogniser instead of a device bound recogniser, use the following code block:

```
val detector = FirebaseVision.getInstance().cloudTextRecognizer
detector.processImage(image)
    .addOnSuccessListener { texts ->
        // Task completed successfully
        // ...
    }
    .addOnFailureListener { e ->
        // Task failed with an exception
        // ...
    }
```

Image Labelling

The Image Labelling API can be used to label or identify the objects within an image. The following steps can be used to identify objects in an image using your android application:

1. In your project-level build.gradle file, make sure to include Google's Maven repository in both your buildscript and allprojects sections.

2. Add the dependencies for the ML Kit Android libraries to your module's app-level gradle file, which is usually app/build.gradle:

   ```
   dependencies {
     // ...
     implementation 'com.google.mlkit:image-labeling:16.0.0'
   }
   ```

3. Create input image from the bitmap file (can also use other methods as mentioned on the Google's Developer website: https://developers.google.com/ml-kit/vision/image-labeling/android)

   ```
   val image = InputImage.fromBitmap(bitmap, 0)
   ```

4. Configure and run the image labeller

   ```
   // To use default options:
   val labeler =
          ImageLabeling.getClient(ImageLabelerOptions.DEFAULT_OPTIONS)
   // Or, to set the minimum confidence required:
   // val options = ImageLabelerOptions.Builder()
   //     .setConfidenceThreshold(0.7f)
   //     .build()
   // val labeler = ImageLabeling.getClient(options)

   labeler.process(image)
           .addOnSuccessListener { labels ->
               // Task completed successfully
               // ...
           }
           .addOnFailureListener { e ->
               // Task failed with an exception
   ```

```
            // ...
        }
```

5. Evaluate each label

```
for (label in labels) {
    val text = label.text
    val confidence = label.confidence
    val index = label.index
}
```

Note: The confidence level for a label is a measure of accuracy of label prediction. If the image exactly matches the label image in the ML model, the confidence level is 1.

Optimal confidence level for accurate predictions should be about 0.7

To use cloud based Image Labeller, use the following block of code:

```
val options = FirebaseVisionCloudImageLabelerOptions.Builder()
     .setConfidenceThreshold(0.5f)
     .build()
val labeler = FirebaseVision.getInstance().getCloudImageLabeler(options)
labeler.processImage(image)
    .addOnSuccessListener { labels ->
        // Task completed successfully
        // ...
    }
    .addOnFailureListener { e ->
        // Task failed with an exception
        // ...
    }
```