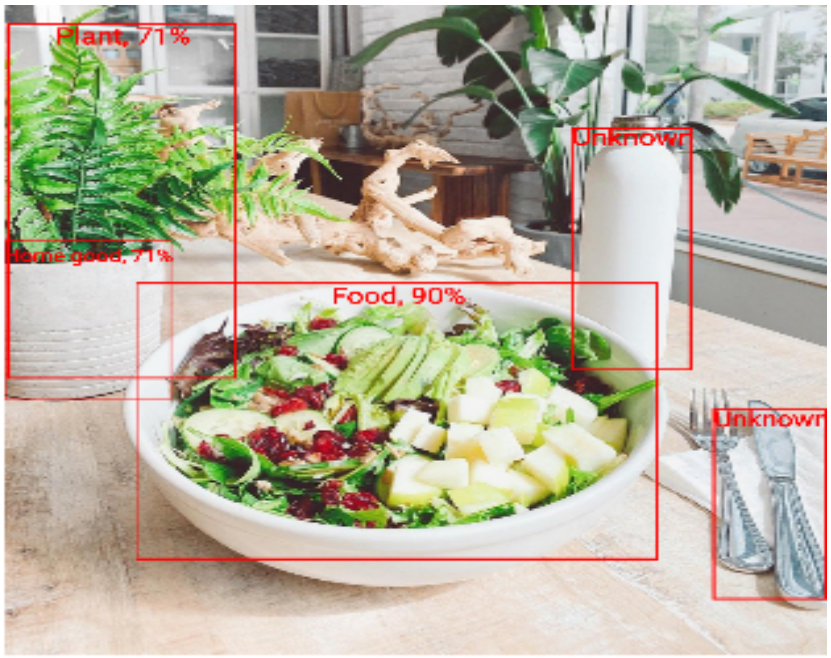
Mobile Engineering: Machine Learning - Laboraufgaben

1) Objekterkennung in Bildern

Laden Sie das Projekt zur Laboraufgabe herunter und öffnen Sie es in Android Studio. Dieses Beispielprojekt soll auf gespeicherten bzw. aufgenommenen Bildern Objekte erkennen, diese mit einer rechteckigen Box umranden und sowohl das vermutete Objekt, als auch eine Übereinstimmungswahrscheinlichkeit ausgeben (siehe Bild).



a) In der MainActivity.kt finden Sie nun die Funktion runObjectDetection(). Diese bekommt ein Bitmap übergeben. Es muss ein ObjectDetector angelegt werden für:

- ein **einzelnes Bild**
- Objekterkennung **mehrerer Objekte**
- mit entsprechender **Klassifizierung**

Dieser verarbeitet das Bild und gibt die Ergebnisse aus.

Die Variable detectedObjects ist ein Mapping der Results: Der Klassifizierungstext der Erkennungsboxen soll im Standardfall “Unknown” sein, sich ansonsten aber aus erkanntem Objekt und Übereinstimmungswahrscheinlichkeit (wie oben im Bild) des entsprechenden Labels zusammensetzen. Die auskommentierten Code-Zeilen dürfen einfach wieder eingefügt werden.

Eine erste Hilfestellung finden Sie unter:
https://developers.google.com/android/reference/com/google/mlkit/vision/objects/package-summary

b) Tauschen Sie die drei eingefügten Bilder unter (/res/drawable) beliebig aus und benennen Sie sie entsprechend. Die Bilder sollten dabei nicht zu groß sein (z.B. 1920x1200). Was erkennt die App gut? Womit hat Sie Probleme? Wie werden die Objekte gelabelt? Probieren Sie verschiedenste Testbilder aus und halten Sie Ihre Beobachtungen fest.

2) Verwendung eines Custom Models

a) Ein geeignetes Modell laden

Suchen Sie auf dem TFHub (*https://tfhub.dev/ml-kit/collections/image-classification/*)
ein anderes geeignetes Modell für unser Laborprojekt. Für die Musterlösung dieser Aufgabe wurde Inception v4 genutzt, haben Sie aber Mut zum Experimentieren!

Achtung: Achten Sie beim Download eines Models darauf, die Metadata-Version zu verwenden!

b) Anpassung build.gradle, Einbindung des Modells

Nach abgeschlossenem Download muss zunächst die **build.gradle** Datei angepasst werden. Statt der object-detection dependency wird nun die *object-detection-custom* dependency benötigt (Version 17). Außerdem wird folgende Option benötigt, um die Komprimierung des Modells zu verhindern:

```
android {
    // ...
    aaptOptions {
        noCompress "tflite"
        // or noCompress "lite"
    }
}
```

Das geladene Modell (.tflite Datei) muss in den Ordner *app/assets* gelegt werden, welcher möglicherweise zunächst angelegt werden muss:
New → Folder → Assets Folder
Target Source Set: main

c) localModel und ObjectDetectorOptions

Erstellen Sie ein *localModel* in der MainActivity und instanziieren Sie dieses in der onCreate-Funktion mit dem LocalModel.Builder(). Dieser erhält mit **.setAssetFilePath(***filename***) und .build()** außerdem den Pfad zum custom-model und die build-Anweisung.

Abschließend müssen die an den objectDetector übergebenen Optionen (*options*) in der **runObjectDetection**-Funktion angepasst werden. Dem **.Builder() der options** wird dafür nun einfach das *localModel* übergeben.

d) Modell testen und Ergebnisse dokumentieren

Halten Sie fest, wie genau (generische Überkategorie oder exakte Objektzuweisung?) und wie sicher ihr ausgewähltes Modell die Objekte auf den eingebauten Bildern erkennt. Laden Sie ein zweites Modell herunter. Hierfür muss jetzt nur noch das neue Modell in den Assets-Ordner kopiert und die entsprechende Codezeile mit dem Pfad angepasst werden. Vergleichen Sie die Resultate.