Université des Mascareignes

Faculty of Information and Communication Technology

Licence Technologique en Informatique Appliquée

3ème année

Semestre 5

PMH

TP6

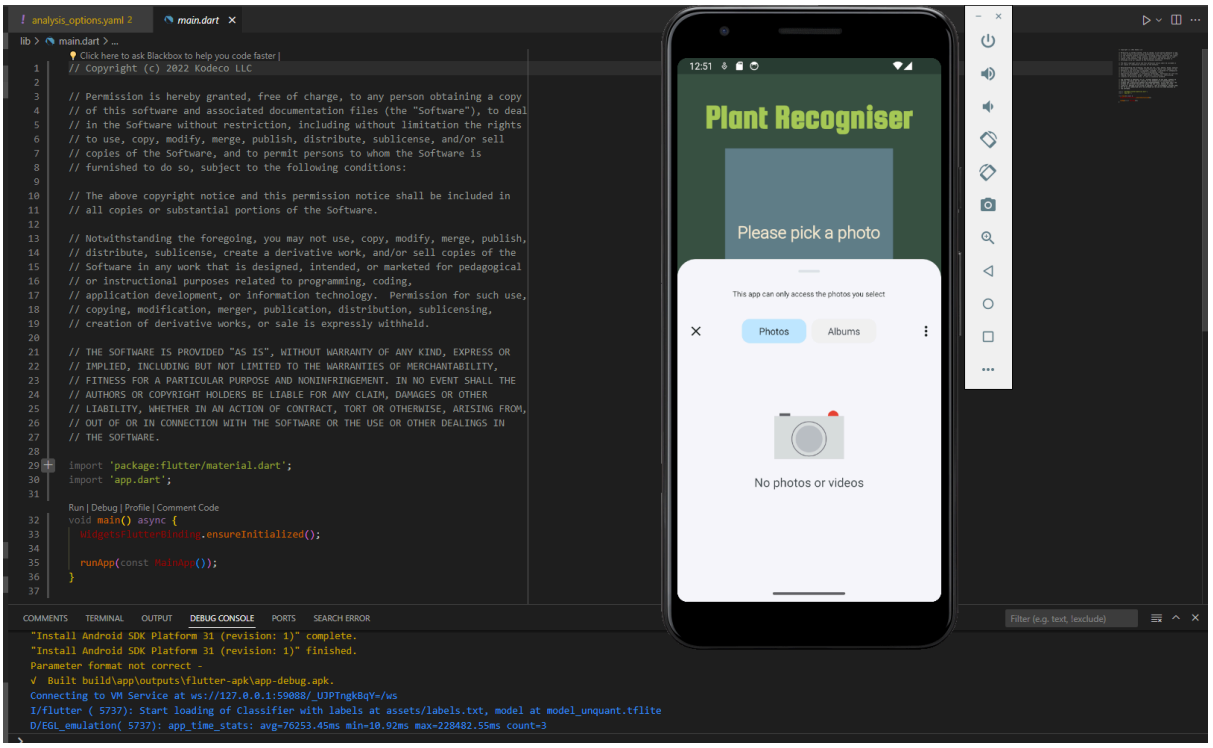
Par : Ken Addison Chan Yin Shin **THIBAUD**

Destinataire : MR. S. Beehary

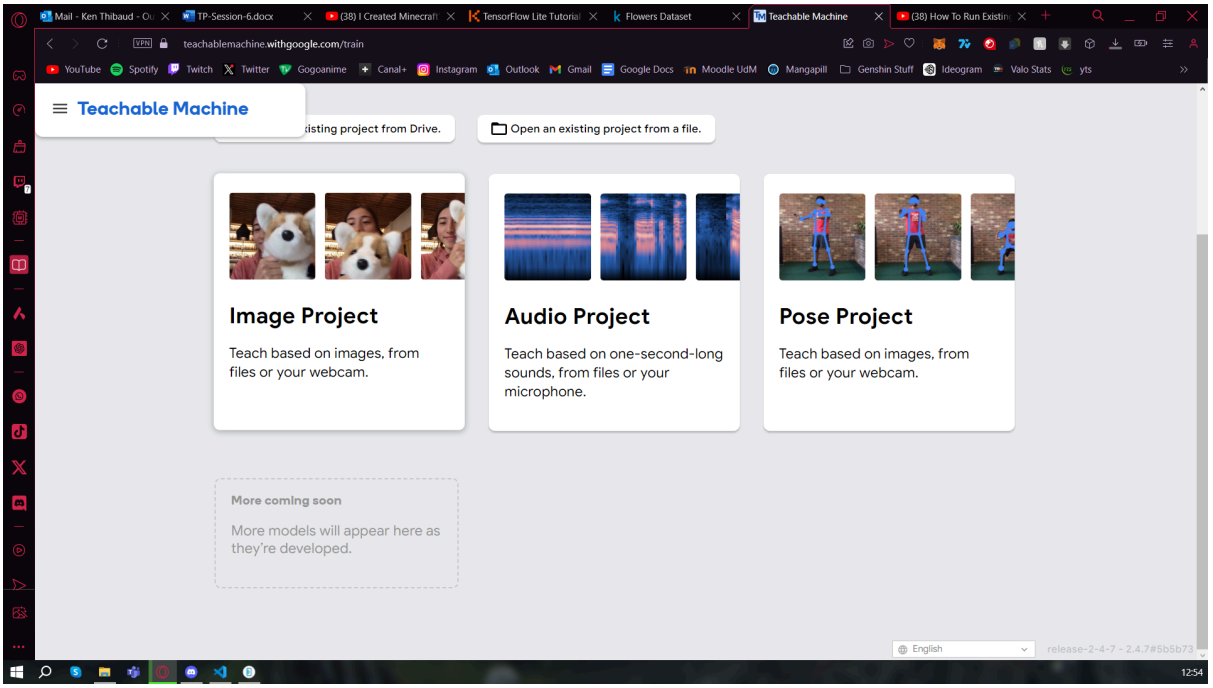
Date: 03/02/2024

Task 1

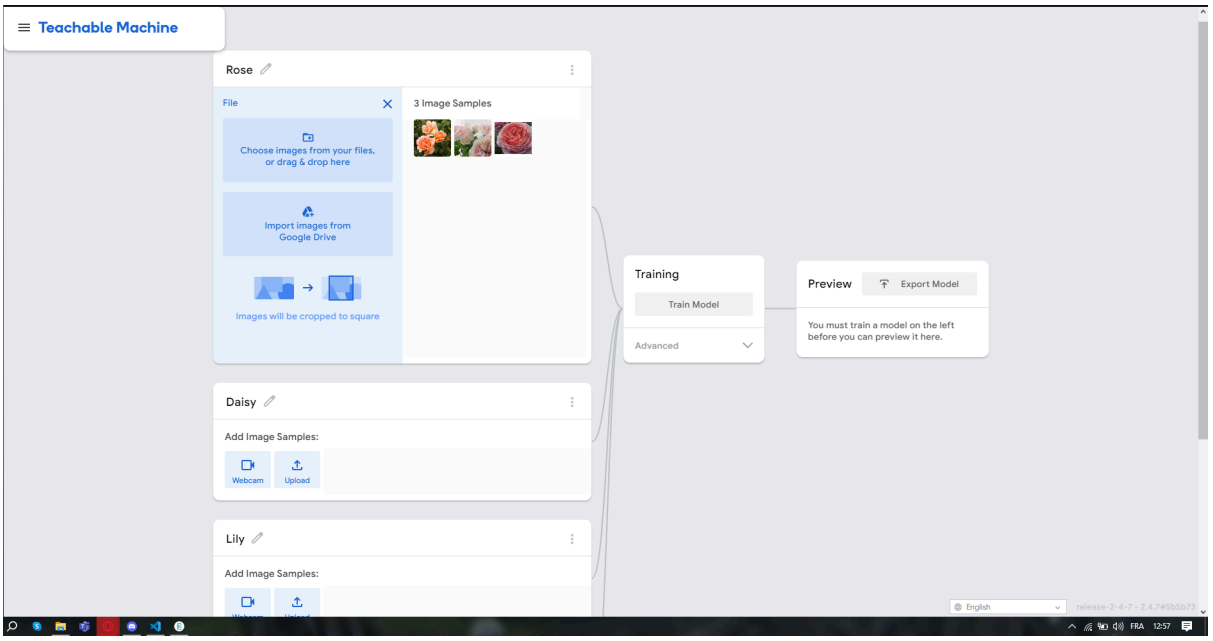
Testing app in VSC



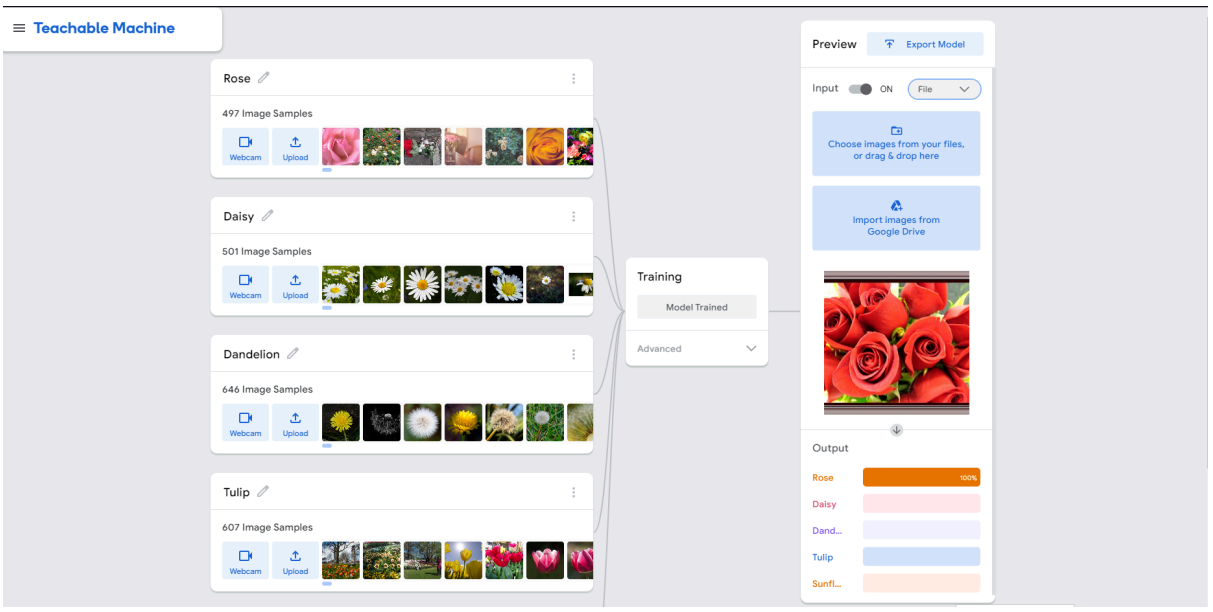
Creating dataset for ML model in teachable



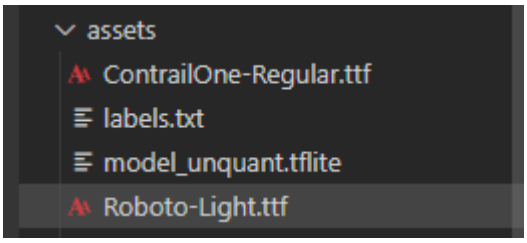
Samples from kaggle link into the teachable machine with corresponding classes



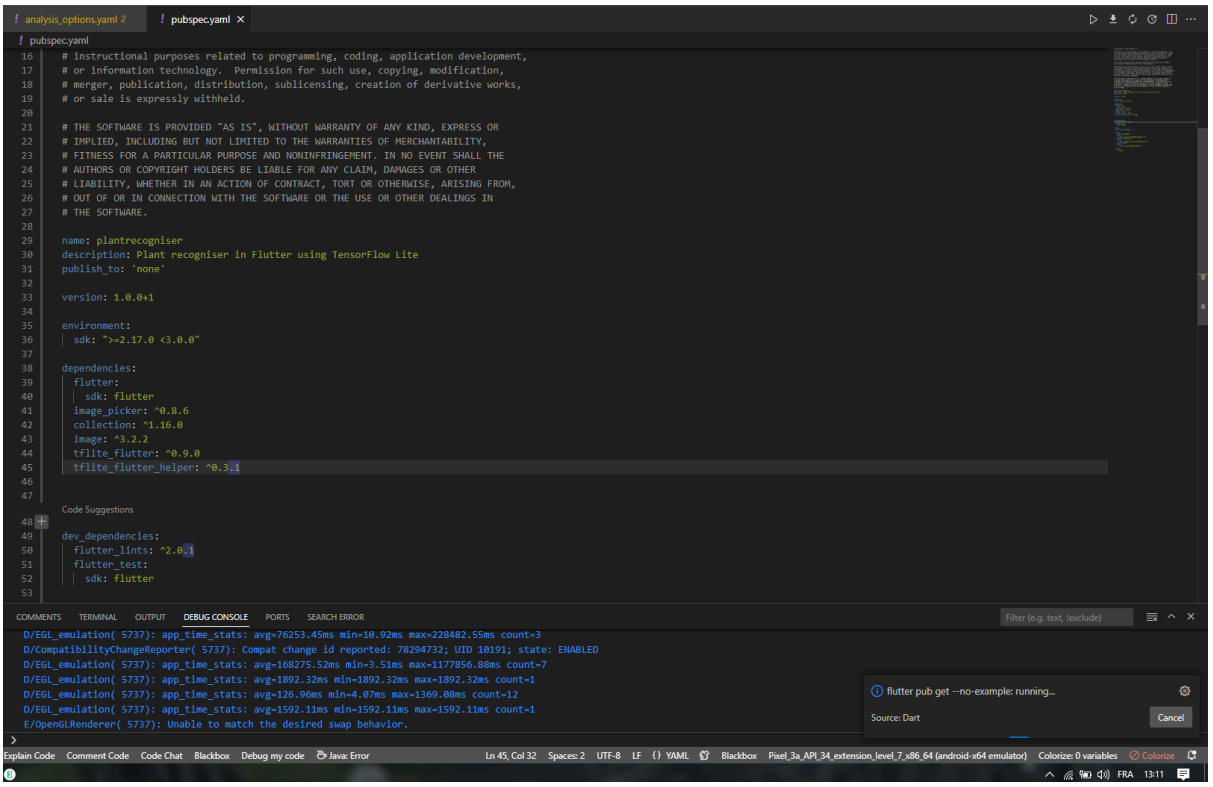
Testing the model



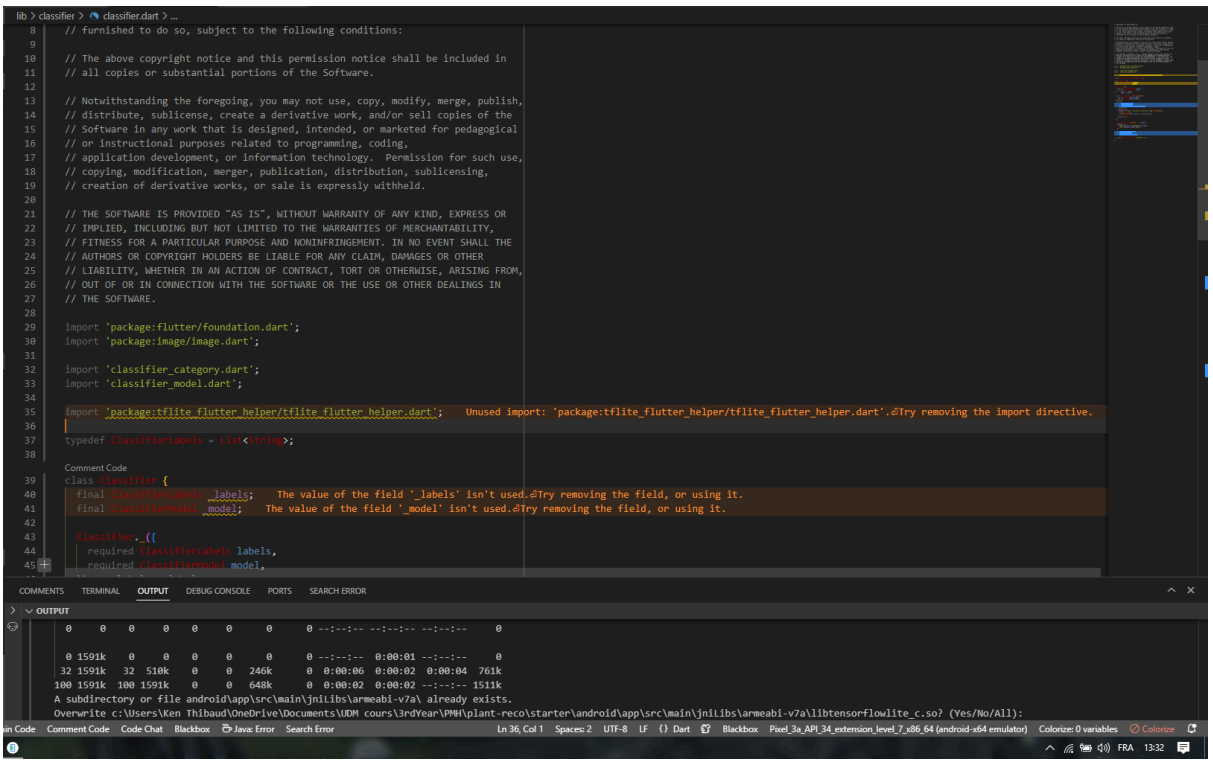
Model and labels that were created were moved to the assets folder of the project



Editing code to be able to use ML model with app



Importing classifier



Code for label classification

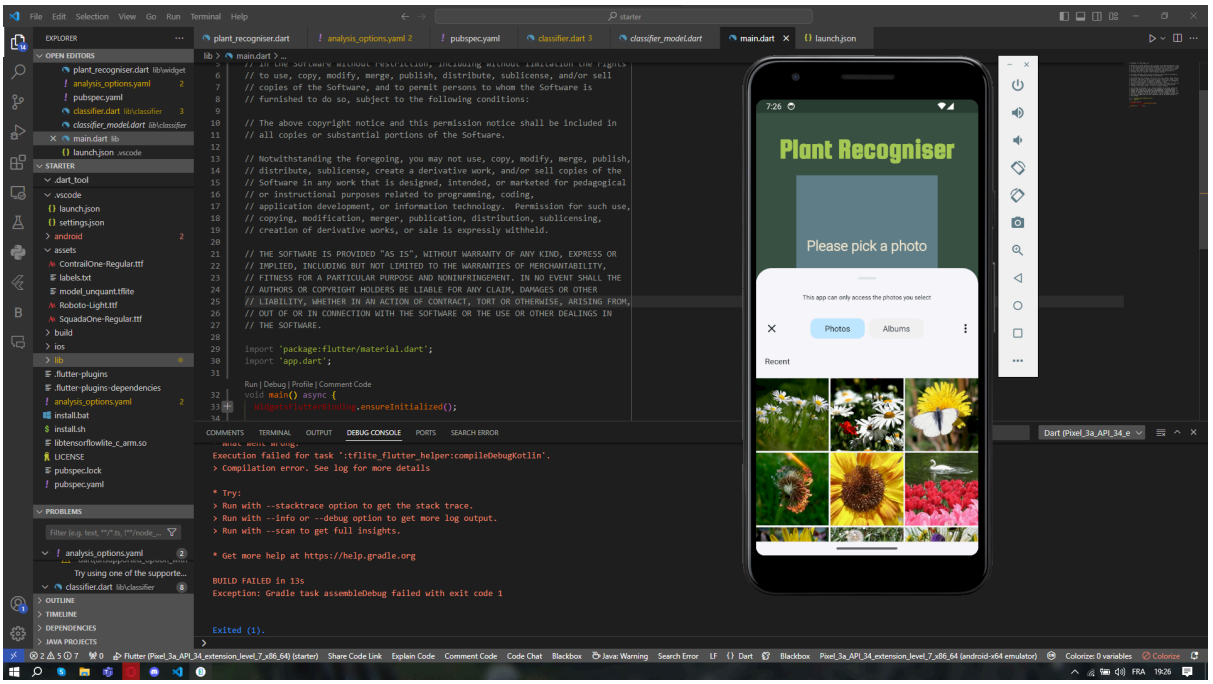
```dart
      return null;
    } catch (e) {
      debugPrint('Can\'t initialize Classifier: ${e.toString()}');
      if (e is Error) {
        debugPrintStack(stackTrace: e.stackTrace);
      }
      return null;
    }
  }

  ClassifierCategory predict(Image image) {
    debugPrint(
      'Image: ${image.width}x${image.height}, '
      'size: ${image.length} bytes',
    );

    // TODO: _preProcessInput    TODO: _preProcessInput
    // TODO: run TF Lite    TODO: run TF Lite
    // TODO: _postProcessOutput    TODO: _postProcessOutput

    return ClassifierCategory('Unknown', 0);
  }

  static Future<ClassifierLabels> _loadLabels(String labelsFileName) async {
    // #1
    final rawLabels = await FileUtil.loadLabels(labelsFileName);

    // #2
    final labels = rawLabels
        .map((label) => label.substring(label.indexOf(' ')).trim())
        .toList();

    debugPrint('Labels: $labels');
    return labels;
  }
}
```

Post process and post process imaging

```dart
    debugPrint('Top category: $topResult');

    return topResult;
  }

  List<ClassifierCategory> _postProcessOutput(TensorBuffer outputBuffer) {
    final probabilityProcessor = TensorProcessorBuilder().build();

    probabilityProcessor.process(outputBuffer);

    final labelledResult = TensorLabel.fromList(_labels, outputBuffer);

    final categoryList = <ClassifierCategory>[];
    labelledResult.getMapWithFloatValue().forEach((key, value) {
      final category = ClassifierCategory(key, value);
      categoryList.add(category);
      debugPrint('label: ${category.label}, score: ${category.score}');
    });
    categoryList.sort((a, b) => (b.score > a.score ? 1 : -1));

    return categoryList;
  }

  TensorImage _preProcessInput(Image image) {
    // #1
    final inputTensor = TensorImage(_model.inputType);
    inputTensor.loadImage(image);

    // #2
    final minLength = min(inputTensor.height, inputTensor.width);
    final cropOp = ResizeWithCropOrPadOp(minLength, minLength);

    // #3
    final shapeLength = _model.inputShape[1];
    final resizeOp = ResizeOp(shapeLength, shapeLength, ResizeMethod.BILINEAR);

    // #4
    final normalizeOp = NormalizeOp(127.5, 127.5);
```

```
/D:/flutter/packages/flutter/lib/src/material/bottom_navigation_bar_theme.dart:5:8: Error: Dart library 'dart:ui' is not available on this platform.
import 'dart:ui' show lerpDouble;
       ^

[Done] exited with code=254 in 14.316 seconds
```

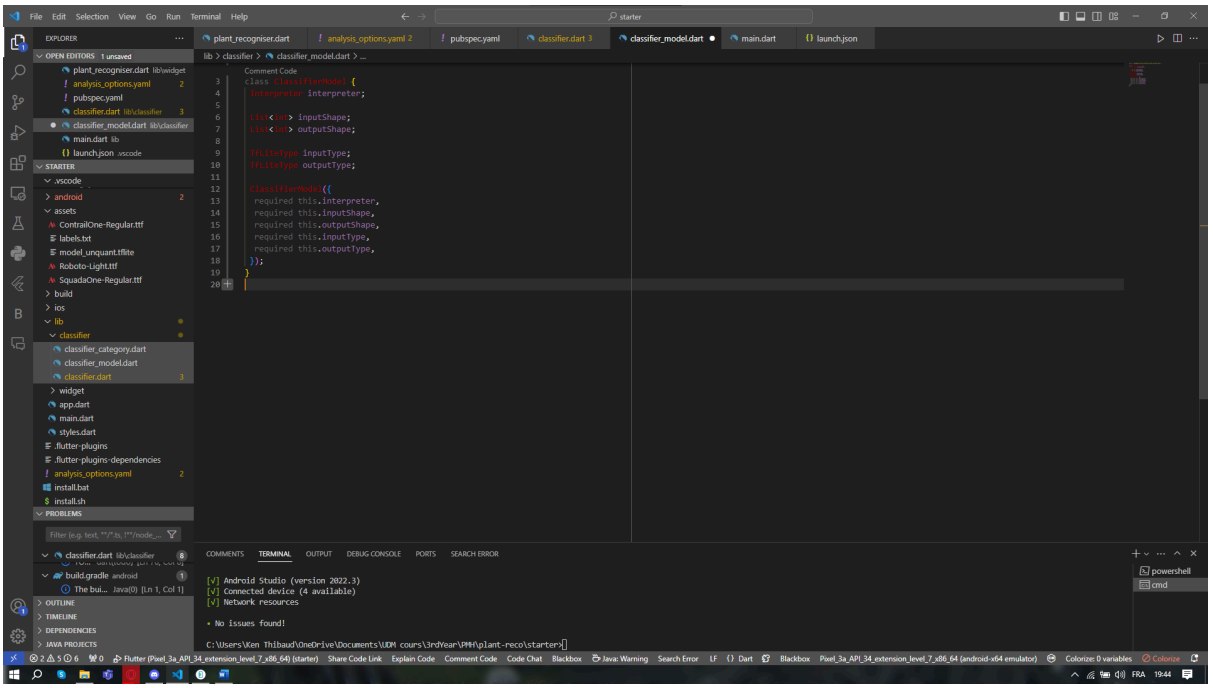
Test images stored in the virtual device



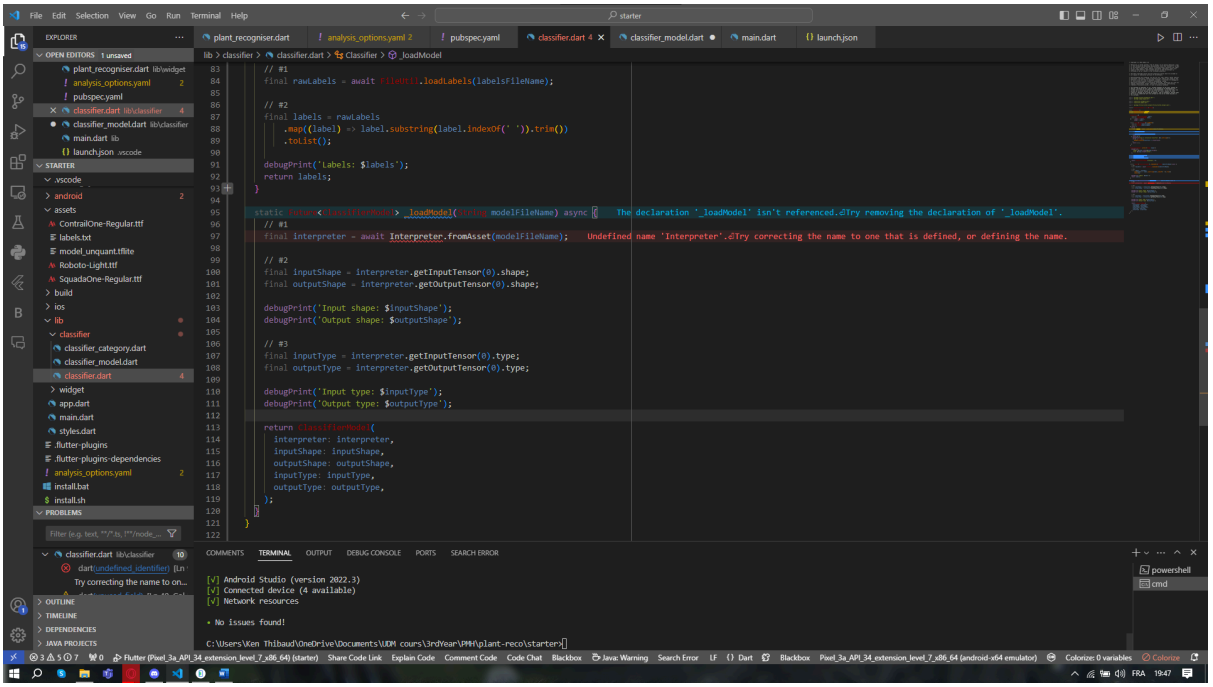
Loading classifier labels

```
static Future<ClassifierLabels> _loadLabels(String labelsFileName) async {
  // #1
  final rawLabels = await FileUtil.loadLabels(labelsFileName);

  // #2
  final labels = rawLabels
      .map((label) => label.substring(label.indexOf(' ')).trim())
      .toList();

  debugPrint('Labels: $labels');
  return labels;
}
}
```

Importing TensorFlow lite model in the classifier model file



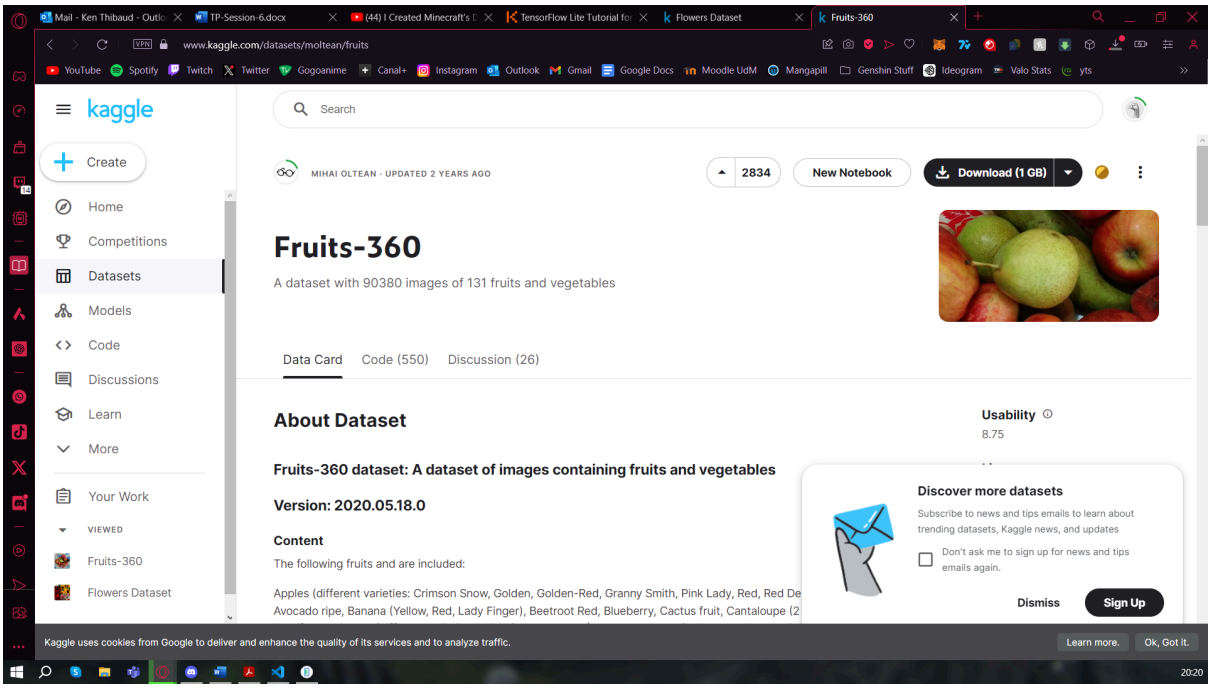
Importing the model from the file



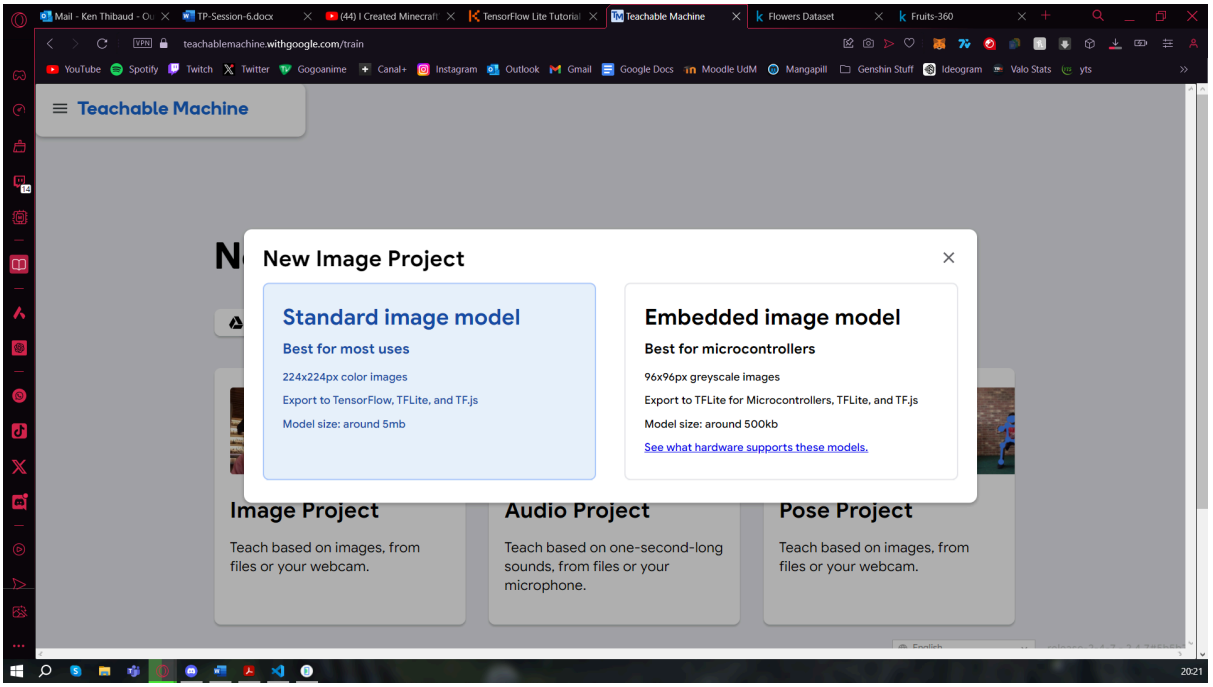
Task 2

This task and task 3 are really similar to task 1

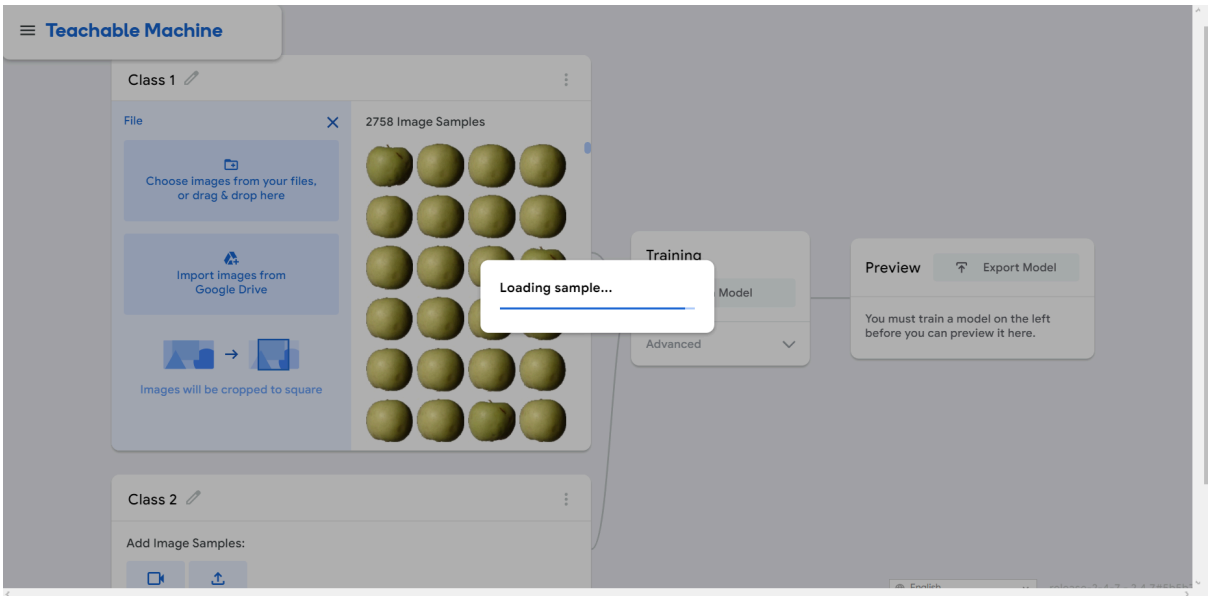
Just import a dataset into teachable machine



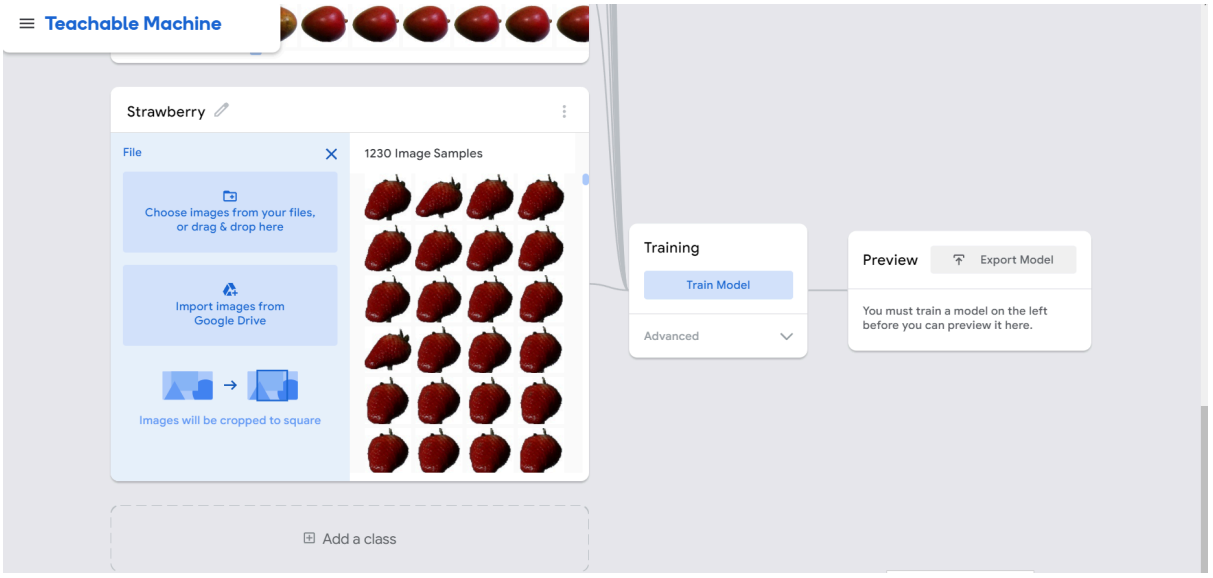
Repeat the step of using standard image model

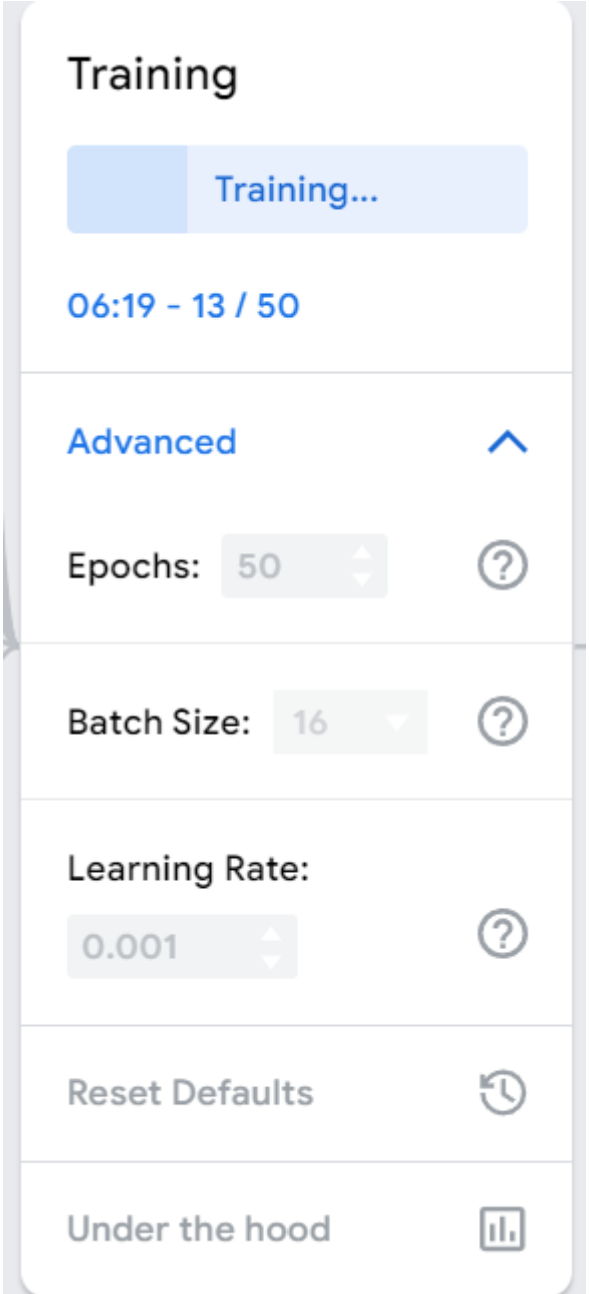


Load the samples into the teachable machine

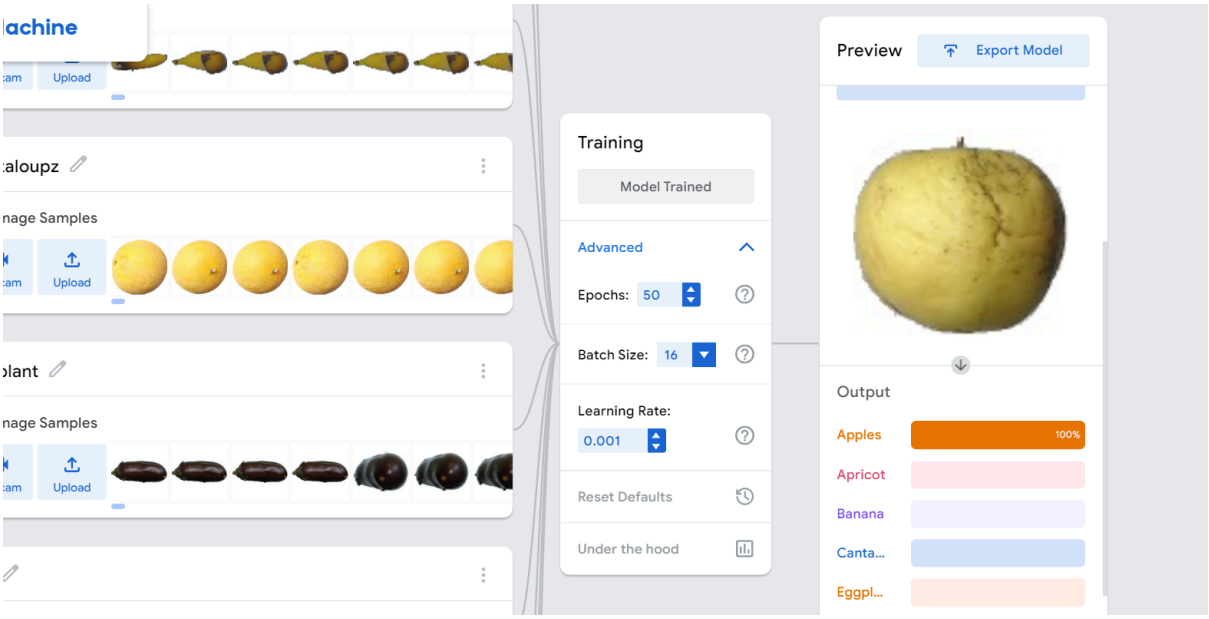


Train the model

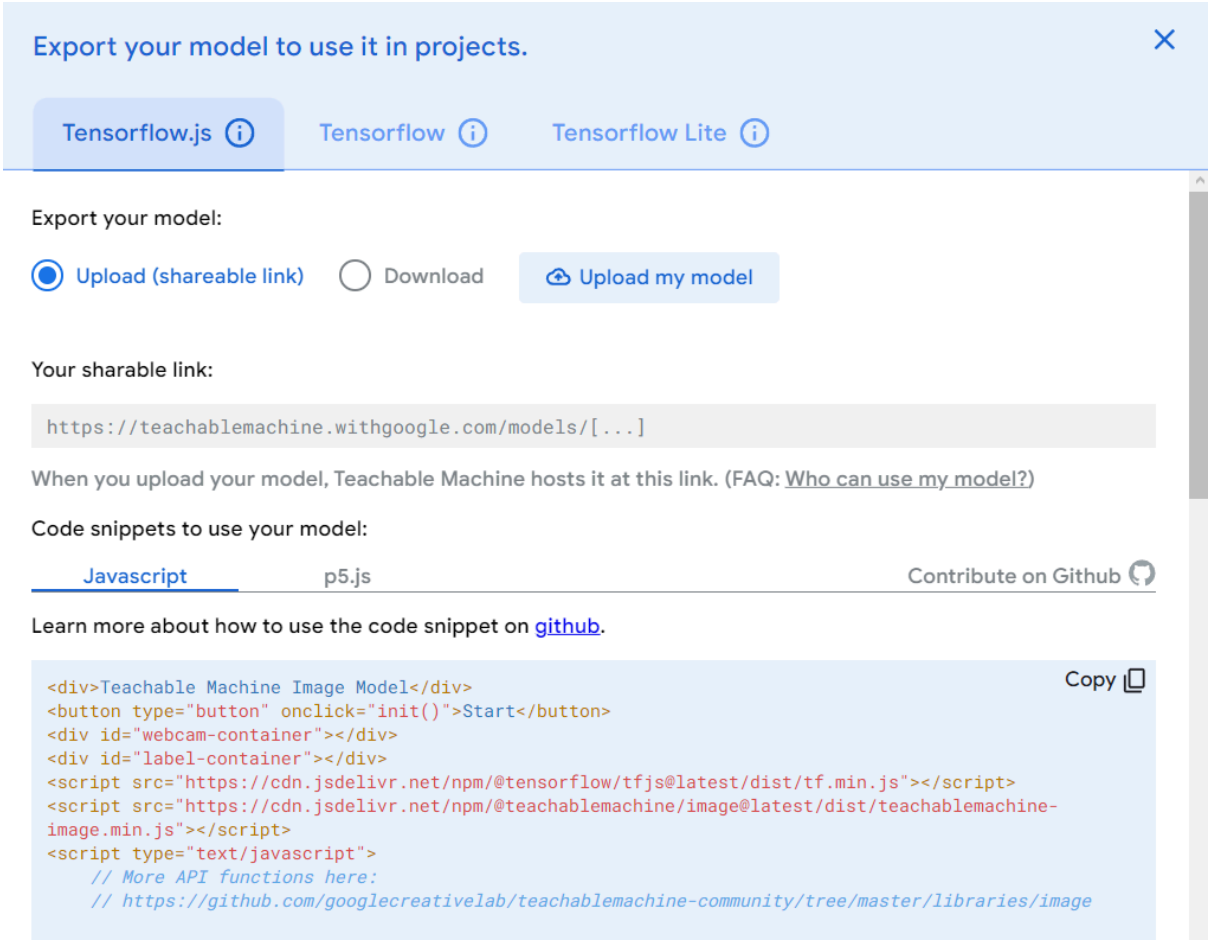



Training
Training...
06:19 - 13 / 50
Advanced
Epochs:
50
Batch Size:
16
Learning Rate:
0.001
Reset Defaults
Under the hood

Test the model with samples



Export labels and model



Then do the same manipulations as for Task 1