Simple Image classification Android app based on ML with Google’s Teachable Machine

Overview –

Our app take snaps of fruits from camera or phone’s gallery and classify them into fresh or rotten fruits of three categories ,namely Banana , Mango and Orange. If none matches it raises an appropriate message.

Prerequisites –

1.Basic programming language knowledge

Technologies Used –

**1.Flutter**

Flutter is Google's SDK for crafting beautiful, fast user experiences for mobile, web, and desktop from a single codebase. Flutter works with existing code, is used by developers and organizations around the world, and is free and open source.

## **2.TensorFlow Lite**

**TensorFlow Lite** is a set of tools that enables on-device machine learning by helping developers run their models on mobile, embedded, and IoT devices.

## **3.Google’s Teachable Machine**

Using [Teachable Machine](https://teachablemachine.withgoogle.com/) from Google, we can develop our own custom ML model using some of our own images without any ML code !! You can train models on images, and poses, sounds, and then can use the trained models in your own projects.

The Recipe -

1. Building the ML Model
2. Head off to <https://teachablemachine.withgoogle.com/> and open a new image project , and name the classes as per the categories you want to classify you images into , for our purpose ,Fresh Oranges ,Rotten Oranges ,Fresh Banana ,Raw Mango , Ripen Mango etc.
3. Upload the respective images under each class from your local repository or from my github repo.
4. Press the TRAIN MODEL button and wait until the process finishes ,Test your model in the preview pane ,If performance is below expectations try twitching the params under Advanced Model Training sections .
5. Once satisfied export the model to tflite floating point model and download it.

2.Initial Setups

i) Import tflite and Image Picker flutter packages from pub.dev and include them under dependencies in your pubspec.yaml file .

ii)Create a new folder named assets in lib directory and paste the two model files that you downloaded earlier .

iii)Head back to pubspec .yaml once again and under assets include the relative path of assets folder. Run pub get form terminal to activate the dependencies and assets.

iv)Import the installed packages in your main.dart file.

v)We declare the following vars inside our app’s state

- \_output var of type list to receive the classification data ;

- \_image var of type File image to get the uploaded image.

- \_loading of type bool to show loading status .

- op object of class ImagePicker to facilitate image capturing .

vi)Below snippet calls the async loadModel method which initiates the ML model when application is started .

void initState() {  
 super.initState();  
 \_loading = true;  
  
 loadModel().then((value) {  
 setState(() {  
 \_loading = false;  
 });  
 });  
}

vii)Below snippet implements the async call back function that when called return an image from gallery .

pickImageGAL() async {  
 PickedFile image = await ip.getImage(source: ImageSource.gallery);  
 if (image == null) return null;  
 setState(() {  
 \_loading = true;  
 \_image = File(image.path);  
 });  
 classifyImage(\_image);  
}

vii)Below snippet implements the async call back function that when called return an image from phone’s camera .

pickImageCAM() async {  
 PickedFile image = await ip.getImage(source: ImageSource.camera);  
 if (image == null) return null;  
 setState(() {  
 \_loading = true;  
 \_image = File(image.path);  
 });  
 classifyImage(\_image);  
}

vii)Below snippet implements the async call back function ClassifyImage that when called feeds the ML model with the image and returns the classification data.

classifyImage(File image) async {  
 var output = await Tflite.*runModelOnImage*(  
 path: image.path,  
 numResults: 2,  
 threshold: 0.8,  
 imageMean: 127.5,  
 imageStd: 127.5,  
 );  
 setState(() {  
 \_loading = false;  
 \_outputs = output;  
  
 });  
}

vii)Below snippet unmounts the model from widget tree and frees the occupied resources when classification is over.

@override  
void dispose() {  
 Tflite.*close*();  
 super.dispose();  
}

3.Constructing the app’s UI

1. Layout

App Bar

A

P

P

B

O

D

Y

COLUMN

WIDGET

ROW WIDGET

Gallery Launcher

Camera Launcher

Result Display Widget

Image Preview Widget

Loading Progress Indicator

**Brief description of Workings;**

When the app launches , it runs the root app called MyApp , which is itself a stateful widget meaning it can remember its state of variables and properties. The app initialises its state and loads the ML model till then setting the \_loading param true ,which renders the circular progress indicator on screen.

The MyApp widget implements the Scaffold ,Which includes the app’s Appbar (Where the title of app goes) and Body (Where the content of app resides). The App body also includes two Raised Buttons which upon pressing ,calls their respective async callback functions and changes state of the root app causing it to re-build its widget tree with updated parameters ,In our case with updated variable that contains the image(\_image object) and data about the classification of image(\_outputs).Which are ultimately rendered on to the screen .We also implement a circular indeterminate progress indicator as an asthetic makeup to our app.

3.Connecting the dots

I)Hook the PickImageGAL and PickImageCAM to respective UI buttons and launch the app on emulator or phone.

FINAL SHOWCASE