UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
GUIA DE PRACTICA DE LABORATORIO /TALLER

CARRERA: Ingeniería en Sistemas e informática	**GUÍA** No. 01	**TIEMPO ESTIMADO:** 1h y 20 min.
ASIGNATURA: Programación Móvil NRC: 6112	**FECHA DE ELABORACION:** 29-Junlioo-2020 **SEMESTRE**: Mayo 2020 – Septiembre 2020	
TÍTULO: Reconocedor de personas con portan o no mascarilla.	**INTEGRANTES:** Rodríguez Fernando y Torres Anthony	

OBJETIVO

Desarrollar una aplicación móvil para sistemas Android y iOS en Flutter para detectar en tiempo real a las personas que portan o no mascarilla a través del uso de técnicas de Machine Learning.

INSTRUCCIONES

i) Revisar la bibliografía de Machine Learning.
ii) Entender como funcionan la clasificación de redes neuronales.
iii) Recolectar fotos de personas con y sin mascarilla.
iv) Entrenar un modelo en TFLite.
v) Crear un proyecto en Flutter.
vi) Acceder a la cámara en Flutter.

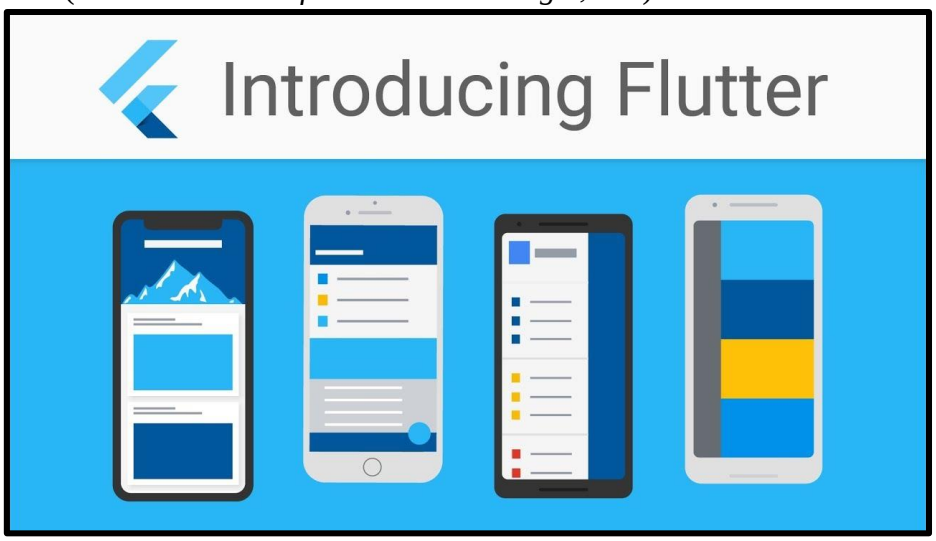
ACTIVIDADES

1. **Ubicación de recursos**
   i) Formar grupos de máximo 2 personas.
   ii) Instalar Flutter y Visual Code.
   iii) Recolectar fotos de personas que utilicen y no utilicen mascarillas.
   iv) Instalar GitHub.
   v) Cuenta en TFlite.

2. Marco Teórico

2.1. Flutter

Flutter es el framework Open Source desarrollado por Google para crear aplicaciones nativas de alta calidad iOS, Android, Desktop o web, todo ello en un tiempo récord. Flutter te permite desarrollar aplicaciones nativas con rendimientos fantásticos. Que se hace evidente con funciones como Hot Reload, con la que puedes hacer cambios en el código y ver el resultado en el emulador en tiempo real, mientras la app sigue en marcha(*Flutter. El nuevo framework de Google*, s. f.).



Flutter puede ayudarte a crear aplicaciones con un buen resultado de UI y UX en un tiempo récord con beneficios tales como:

- Desarrollo de alta velocidad: Con Hot Reload en milisegundos puedes dar vida a tu aplicación. Con Flutter tienes a tu alcance un amplio conjunto de widgets totalmente personalizables para construir interfaces nativas en minutos.
- Diseños adaptados y flexibles: Flutter rápidamente libera funciones con un enfoque claro en las experiencias nativas del usuario final, te permite adaptarte a cada escenario con facilidad. La arquitectura en capas te permite

una personalización completa, lo que resulta en una renderización increíblemente rápida, diseños adaptados y flexibles.

- Rendimiento nativo: Los widgets de Flutter incorporan todas las diferencias críticas de cada plataforma. Algunas como el scrolling, navegación, iconos y fuentes para proporcionar un completo rendimiento nativo en cada plataforma, ya sea iOS o Android.

Desde el lanzamiento alfa en 2017, Google brinda, con la ayuda de la comunidad múltiples funciones nativas. Como soporte de lector de pantalla y otras funciones de accesibilidad, texto de derecha a izquierda, multi idioma, compatibilidad con nuevos dispositivos como iPhone X e iOS 12 y mucho más.

A diferencia de otros frameworks similares, como Ionic, una aplicación Flutter compilada no contiene ningún código interpretado o WebView, al igual que React Native. Todo el código se compila a binario de forma nativa para la plataforma en la que se está desarrollando. Todo esto es posible gracias a Dart, el lenguaje utilizado por Flutter.

2.2. Red Neuronal

Las redes neuronales artificiales (ANN) son un sistema de aprendizaje supervisado construido con una gran cantidad de elementos simples, llamados neuronas o perceptrones. Cada neurona puede tomar decisiones simples y las transmite a otras neuronas, organizadas en capas interconectadas. En conjunto, la red neuronal puede emular casi cualquier función y responder prácticamente cualquier pregunta, con suficientes muestras de entrenamiento y potencia de cálculo. Una red neuronal "superficial" tiene solo tres capas de neuronas:

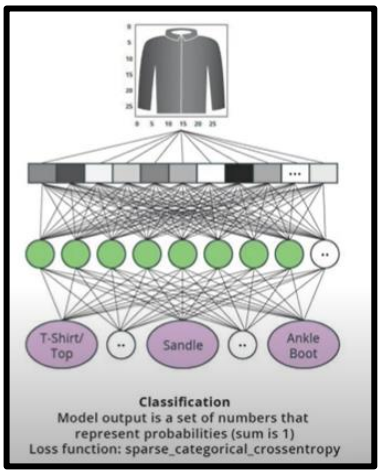
- Una capa de entrada que acepta las variables independientes o entradas del modelo.
- Una capa oculta
- Una capa de salida que genera predicciones.

Una red neuronal profunda (DNN) tiene una estructura similar, pero tiene dos o más "capas ocultas" de neuronas que procesan las entradas. Goodfellow, Bengio y Courville demostraron que, si bien las redes neuronales superficiales pueden abordar problemas complejos, las redes de aprendizaje profundo son más precisas y mejoran en precisión a medida que se agregan más capas de neuronas. Las capas adicionales

son útiles hasta un límite de 9-10, después de lo cual su poder predictivo comienza a disminuir. En la actualidad, la mayoría de los modelos e implementaciones de redes neuronales utilizan una red profunda de entre 3 y 10 capas de neuronas.

2.2.1. Clasificación con redes neuronales

La fortaleza única de una red neuronal es su capacidad para crear dinámicamente funciones de predicción complejas y resolver problemas de clasificación de una manera que emula el pensamiento humano. Para ciertos problemas de clasificación, las redes neuronales pueden proporcionar un rendimiento mejorado en comparación con otros algoritmos. Sin embargo, debido a que las redes neuronales son más intensivas en computación y más complejas de configurar, pueden ser excesivas en muchos casos(*Complete Guide to Artificial Neural Network Concepts & Models*, s. f.).



Tipos de algoritmos de clasificación

Para comprender la clasificación con redes neuronales, cubramos algunos otros algoritmos de clasificación comunes. Algunos algoritmos son binarios, proporcionando una decisión de sí / no, mientras que otros son de clases múltiples, lo que le permite clasificar una entrada en varias categorías.

Regresión logística (binaria): analiza un conjunto de puntos de datos y encuentra el modelo que mejor se ajusta para describirlos. Fácil de implementar

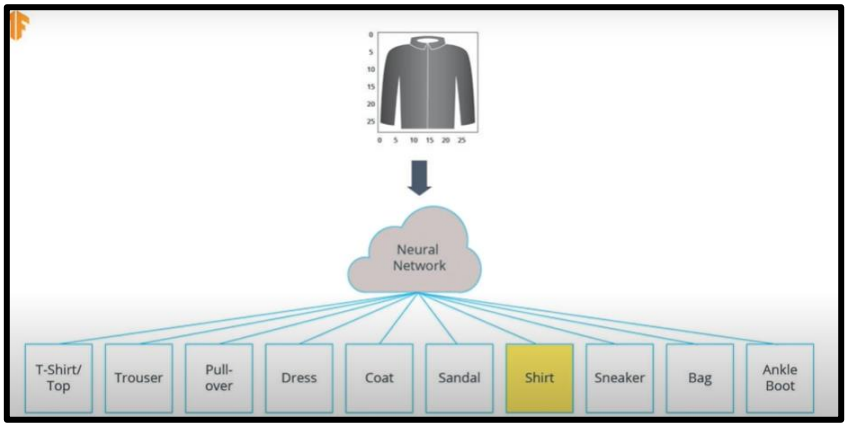
y muy eficaz para las variables de entrada que son bien conocidas y están estrechamente relacionadas con el resultado.

Árbol de decisión (multiclase): clasifica utilizando una estructura de árbol con reglas si-entonces, ejecutando la entrada a través de una serie de decisiones hasta que alcanza una condición de terminación. Capaz de modelar procesos de decisión complejos y es muy intuitivo, pero puede sobreajustar fácilmente los datos.

Bosque aleatorio (multiclase): conjunto de árboles de decisión, con selección automática del árbol con mejor rendimiento. Proporciona la fuerza del algoritmo del árbol de decisión sin el problema de sobreajuste.

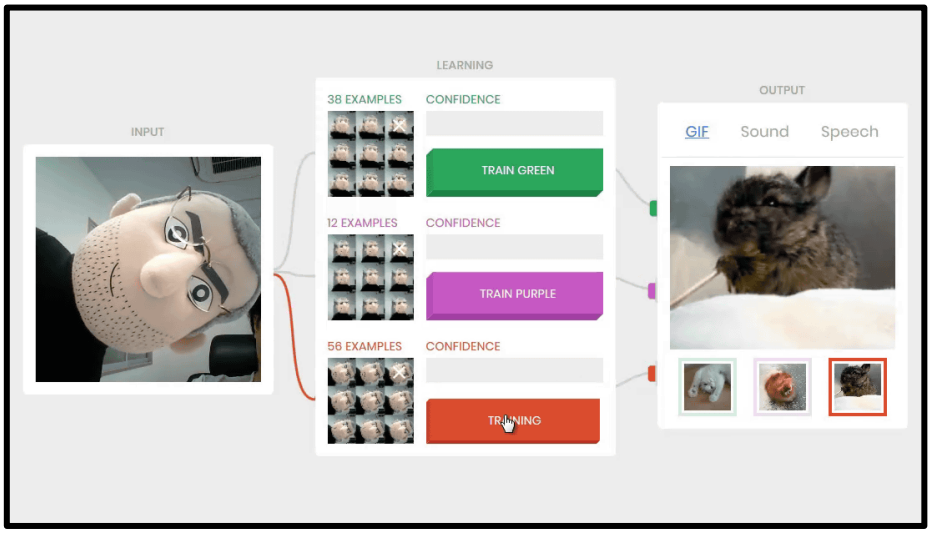
Clasificador Naive Bayes (multiclase): un clasificador basado en probabilidades. Calcula la probabilidad de que cada punto de datos exista en cada una de las categorías de destino. Simple de implementar y preciso para una gran cantidad de problemas, pero sensible al conjunto de categorías seleccionadas.

k-Vecino más cercano (multiclase): clasifica cada punto de datos analizando sus vecinos más cercanos entre los ejemplos de entrenamiento. Simple de implementar y comprender, eficaz para muchos problemas, especialmente aquellos con baja dimensionalidad. Proporciona una menor precisión en comparación con los algoritmos supervisados y es computacionalmente intensivo.



2.3. Teachable Machine

Es una plataforma de Inteligencia Artificial que permite al usuario entrenar de manera independiente a una red neuronal, mientras que el sistema puede reaccionar a imágenes conocidas con sonidos o animaciones GIF. La versión de demostración está disponible en el sitio web del proyecto(*Teachable Machine, la inteligencia artificial de Google, te responde con un GIF*, s. f.).



Estas son algunas de las características más destacadas de la Teachable Machine actualizada(*Google actualiza Teachable Machine con una inteligencia artificial*, s. f.):

- Entrenar un modelo en datos de imagen.
- Entrenar un modelo en datos de audio.
- Entrenar un modelo en datos de pose (como en Posenet).
- Cargar sus propios conjuntos de datos para entrenar modelos.
- Entrenar más de 3 clases por modelo.
- Deshabilitar clases.
- Guardar su modelo TensorFlow.js.
- Descargar tu modelo.
- Implementar tu modelo para usar en tu propio proyecto (sitios web, aplicaciones, máquinas físicas, etc.)

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
GUIA DE PRACTICA DE LABORATORIO /TALLER

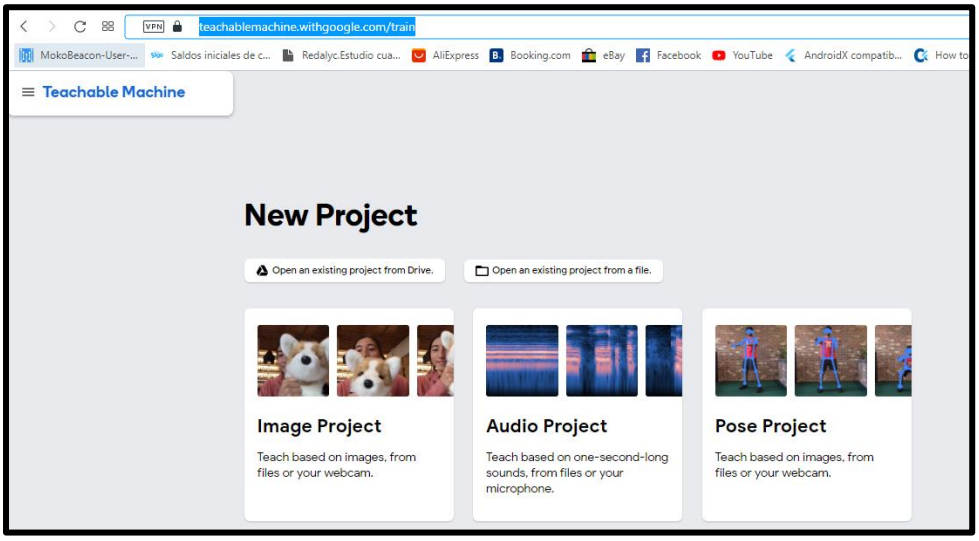
3. Planteamiento del problema

APP para reconocer a las personas que utilizan o no mascarilla

Desarrollar una aplicación móvil utilizando el framework de Flutter, esta aplicación debe reconocer en tiempo real a las personas que utilizan o no mascarilla, para eso se debe utilizar la cámara del dispositivo. Además, cuando una persona sea enfocada debe mostrar el porcentaje de clasificación en una barra, esto va de 0-100%. Para el modelo neuronal se puede usar el más conveniente.

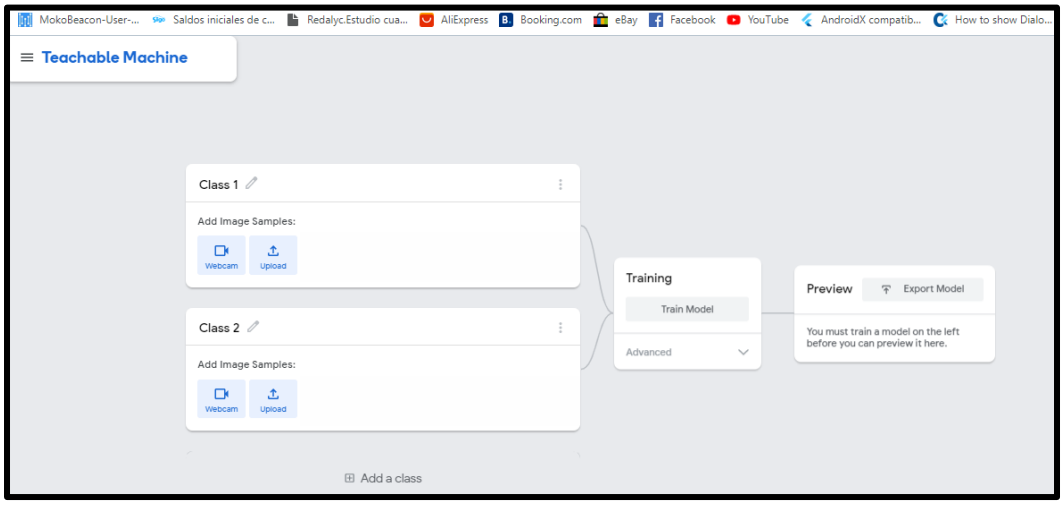
Entrenando a la red neuronal

Dirigirse al siguiente link: https://teachablemachine.withgoogle.com/train
Clic en Image Project

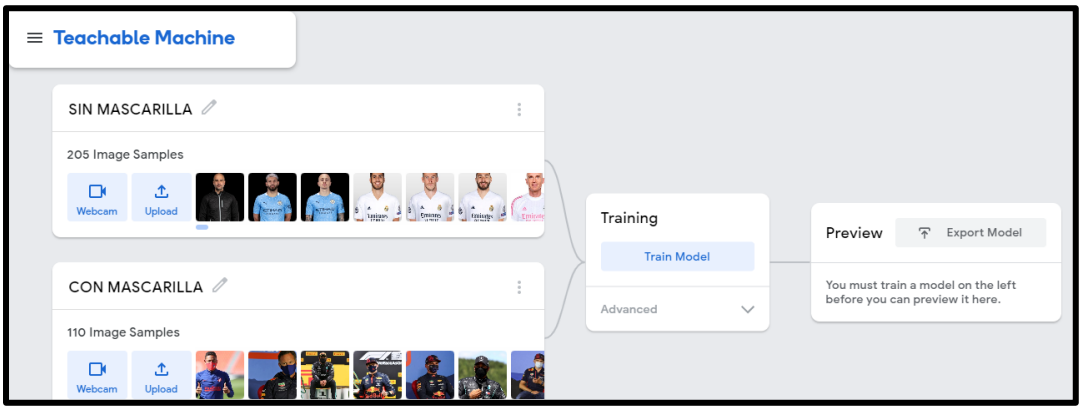


Aparecerá la siguiente página.

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
GUIA DE PRACTICA DE LABORATORIO /TALLER

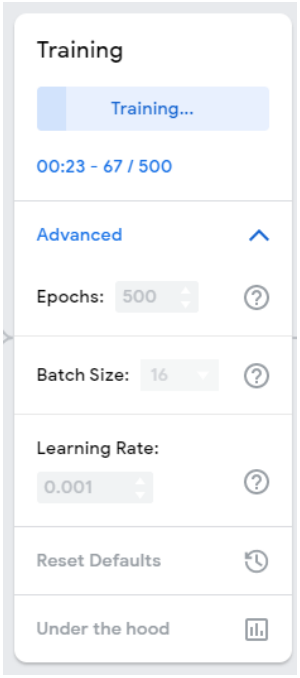


Aquí se colocará las dos clasificaciones que se necesita para este trabajo: SIN MASCARILLA Y CON MASCARIILA. Subir las imágenes, debe ser mayor a 100.



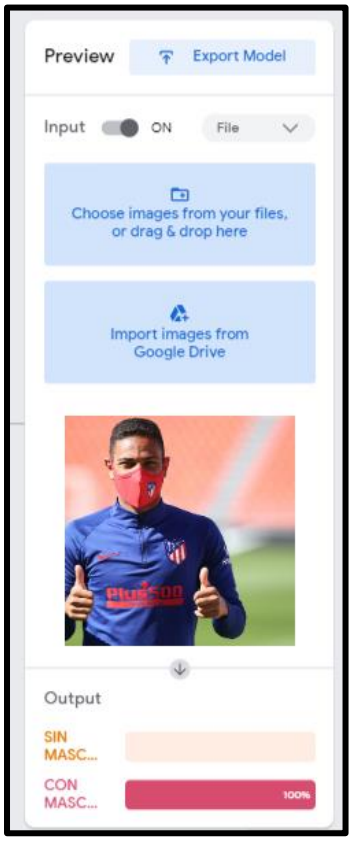
Entrenar el modelo, dar clic en Train Model

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
GUIA DE PRACTICA DE LABORATORIO /TALLER



Probar el modelo

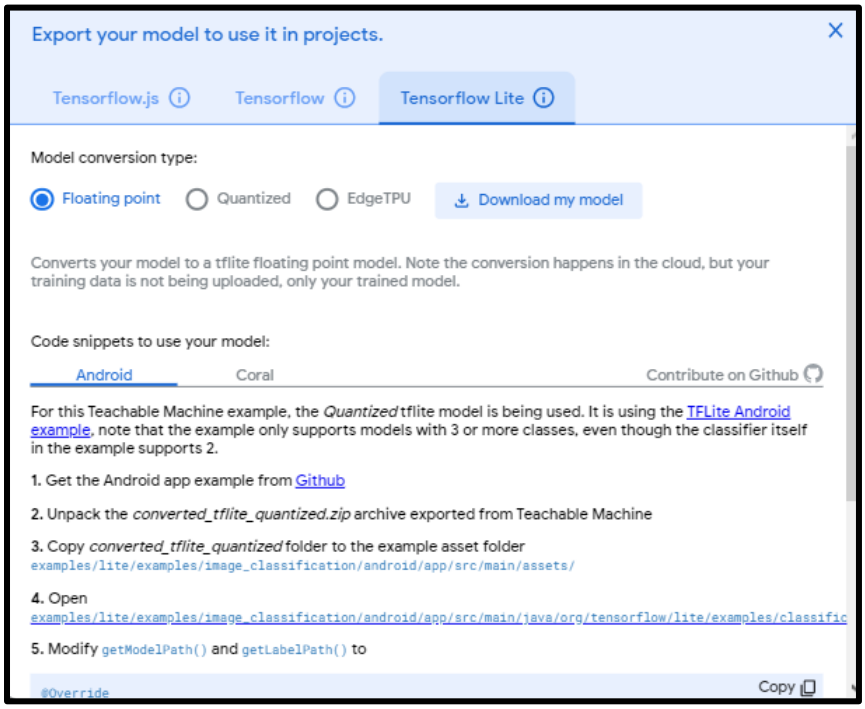
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
GUIA DE PRACTICA DE LABORATORIO /TALLER



Exportar el modelo en TensorflowLite
Se descargará un zip

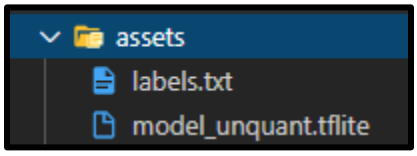
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
GUIA DE PRACTICA DE LABORATORIO /TALLER



Utilizando el modelo en el proyecto

Crear la carpeta assets
Pegar el contenido de nuestro modelo entrenado previamente



Descargando dependencias

Dirigirse al archivo pubspec.yaml

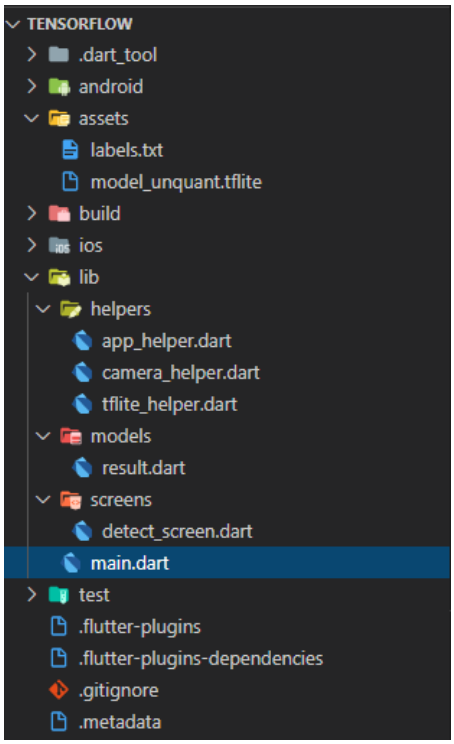
```
dependencies:
  flutter:
    sdk: flutter

  # The following adds the Cuper
  # Use with the CupertinoIcons
  cupertino_icons: ^0.1.2

  tflite: ^1.0.5
  percent_indicator: ^2.1.1+1
  camera: ^0.5.7+4
```

Código

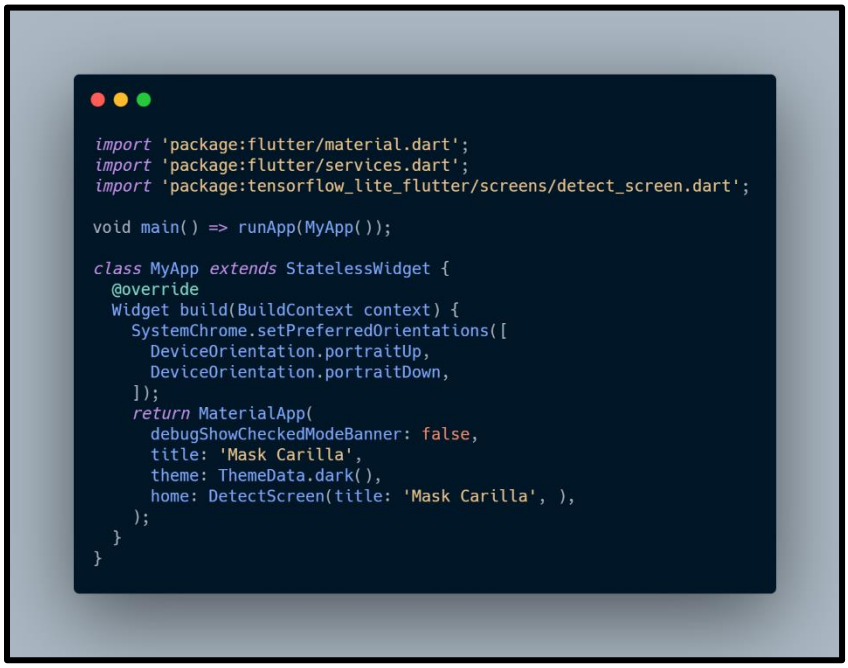
Estructura del proyecto



main.dart

UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
GUIA DE PRACTICA DE LABORATORIO /TALLER



```dart
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:tensorflow_lite_flutter/screens/detect_screen.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    SystemChrome.setPreferredOrientations([
      DeviceOrientation.portraitUp,
      DeviceOrientation.portraitDown,
    ]);
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Mask Carilla',
      theme: ThemeData.dark(),
      home: DetectScreen(title: 'Mask Carilla', ),
    );
  }
}
```

detect_screen.dart

UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
GUIA DE PRACTICA DE LABORATORIO /TALLER

```dart
import 'package:camera/camera.dart';
import 'package:flutter/foundation.dart';
import 'package:flutter/material.dart';
import 'package:percent_indicator/linear_percent_indicator.dart';
import 'package:tensorflow_lite_flutter/helpers/app_helper.dart';
import 'package:tensorflow_lite_flutter/helpers/camera_helper.dart';
import 'package:tensorflow_lite_flutter/helpers/tflite_helper.dart';
import 'package:tensorflow_lite_flutter/models/result.dart';

class DetectScreen extends StatefulWidget {
  DetectScreen({Key key, this.title}) : super(key: key);

  final String title;

  @override
  _DetectScreenPageState createState() => _DetectScreenPageState();
}

class _DetectScreenPageState extends State<DetectScreen>
    with TickerProviderStateMixin {
  AnimationController _colorAnimController;
  Animation _colorTween;

  List<Result> outputs;

  void initState() {
    super.initState();

    //Cargando TFLite Model
    TFLiteHelper.loadModel().then((value) {
      setState(() {
        TFLiteHelper.modelLoaded = true;
      });
    });

    //Camera
    CameraHelper.initializeCamera();

    //Inicar animacion
    _setupAnimation();

    //TFLite Classificacion
    TFLiteHelper.tfLiteResultsController.stream.listen((value) {
      value.forEach((element) {
        _colorAnimController.animateTo(
          element.confidence,
            curve: Curves.bounceIn,
            duration: Duration(milliseconds: 500)
        );
      });

      //Resultados
      outputs = value;

      //Update los resultados en pantalla
      setState(() {

        CameraHelper.isDetecting = false;
      });
    }, onDone: () {

    }, onError: (error) {
      AppHelper.log("listen", error);
    });
  }
```

```dart
@override
Widget build(BuildContext context) {
  double width = MediaQuery.of(context).size.width;
  return Scaffold(
    appBar: AppBar(
      title: Text(widget.title),
      centerTitle: true,
    ),
    body: FutureBuilder<void>(
      future: CameraHelper.initializeControllerFuture,
      builder: (context, snapshot) {
        if (snapshot.connectionState == ConnectionState.done) {
          // Se despliega la camera
          return Stack(
            children: <Widget>[
              CameraPreview(CameraHelper.camera),
              _buildResultsWidget(width, outputs)
            ],
          );
        } else {
          // Otherwise, display a loading indicator.
          return Center(child: CircularProgressIndicator());
        }
      },
    ),
  );
}

/// Metodo para cerrar TFLiteHelper,CameraHelper y AppHelper
@override
void dispose() {
  TFLiteHelper.disposeModel();
  CameraHelper.camera.dispose();
  AppHelper.log("dispose", "Clear resources.");
  super.dispose();
}

/// Widget que construye las barras y el porcentaje de precisión
/// De clasificacion de la imagen
Widget _buildResultsWidget(double width, List<Result> outputs) {
  return Positioned.fill(
    child: Align(
      alignment: Alignment.bottomCenter,
      child: Container(
        height: 200.0,
        width: width,
        color: Colors.blueGrey[900],
        child: outputs != null && outputs.isNotEmpty
            ? ListView.builder(
                itemCount: outputs.length,
                shrinkWrap: true,
                padding: const EdgeInsets.all(20.0),
                itemBuilder: (BuildContext context, int index) {
                  return Column(
                    children: <Widget>[
                      Text(
                        outputs[index].label,
                        style: TextStyle(
                          color: _colorTween.value,
                          fontSize: 20.0,
                        ),
                      ),
                      Text('data'),
                      AnimatedBuilder(
                          animation: _colorAnimController,
                          builder: (context, child) => LinearPercentIndicator(
                                width: width * 0.88,
                                lineHeight: 14.0,
                                percent: outputs[index].confidence,
                                progressColor: _colorTween.value,
                              )),

                      Text(
                        "${(outputs[index].confidence * 100.0).toStringAsFixed(2)} %",
                        style: TextStyle(
                          color: _colorTween.value,
                          fontSize: 16.0,
                        ),
                      ),
                    ],
                  );
                })

            : Center(
                child: Text("Wating for model to detect..",
                    style: TextStyle(
                      color: Colors.black,
                      fontSize: 20.0,
                    ))),
      ),
    ),
  );
}

/// La animacion de las barras de los colores
void _setupAnimation() {
  _colorAnimController =
      AnimationController(vsync: this, duration: Duration(milliseconds: 500));
  _colorTween = ColorTween(begin: Colors.tealAccent[100], end: Colors.tealAccent[400])
      .animate(_colorAnimController);
}
}
```

result.dart

```
class Result {
  double confidence;
  int id;
  String label;

  Result(this.confidence, this.id, this.label);
}
        .animate(_colorAnimController);
  }
}
```

tflite_helper.dart

UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
GUIA DE PRACTICA DE LABORATORIO /TALLER

```dart
import 'dart:async';
import 'package:camera/camera.dart';
import 'package:tensorflow_lite_flutter/models/result.dart';
import 'package:tflite/tflite.dart';
import 'app_helper.dart';

class TFLiteHelper {

  static StreamController<List<Result>> tfLiteResultsController = new
StreamController.broadcast();
  static List<Result> _outputs = List();
  static var modelLoaded = false;


  /// Metodo que carga el modelo realizado en Tflite
  static Future<String> loadModel() async{
    AppHelper.log("loadModel", "Loading model..");

    return Tflite.loadModel(
      model: "assets/model_unquant.tflite",
      labels: "assets/labels.txt",
    );
  }

  static classifyImage(CameraImage image) async {

   /// Metodo que utiliza en tiempo real el modelo generado en Tflite
   /// utlizando la camara para clasificar los resultados
    await Tflite.runModelOnFrame(
            bytesList: image.planes.map((plane) {
              return plane.bytes;

            }).toList(),

            numResults: 2,
            imageHeight:image.height ,
            imageWidth: image.width,
            imageMean: 127.5,
            imageStd:127.5 ,
            threshold: 0.1,
            asynch: true,
            )
        .then((value) {
      if (value.isNotEmpty) {
        AppHelper.log("classifyImage", "Results loaded.
${value.length}");

        //Limpia los resultados anteriores
        _outputs.clear();

        value.forEach((element) {
          _outputs.add(Result(
              element['confidence'], element['index'],
element['label']));

          AppHelper.log("classifyImage",
              "${element['confidence']} , ${element['index']},
${element['label']}");
        });
      }
      /// Ordenar resultados por confianza
      _outputs.sort((a, b) => a.confidence.compareTo(b.confidence));

      /// Envia los resultados a la lista
      tfLiteResultsController.add(_outputs);
    });
  }
  /// Metodo para cerrar tfLiteResultsController
  static void disposeModel(){
    Tflite.close();
    tfLiteResultsController.close();
  }
}
```

camera_helper.dart

```dart
import 'package:camera/camera.dart';
import 'package:flutter/foundation.dart';
import 'package:flutter/material.dart';
import 'package:tensorflow_lite_flutter/helpers/app_helper.dart';
import 'package:tensorflow_lite_flutter/helpers/tflite_helper.dart';

/// Clase que accede a la camara y la inicializa
class CameraHelper {
  static CameraController camera;

  static bool isDetecting = false;
  //Establece el tipo de camera que se va usar, back o front
  static CameraLensDirection _direction = CameraLensDirection.back;
  static Future<void> initializeControllerFuture;

  /// Metodo que devuelve la camara en base al parametro
CameraLensDirection
  static Future<CameraDescription> _getCamera(CameraLensDirection dir)
async {
    return await availableCameras().then(
      (List<CameraDescription> cameras) => cameras.firstWhere(
        (CameraDescription camera) => camera.lensDirection == dir,
      ),
    );
  }

/// Metodo static para inciar la camara
  static void initializeCamera() async {
    AppHelper.log("_initializeCamera", "Initializing camera..");

    camera = CameraController(
        await _getCamera(_direction),
        defaultTargetPlatform == TargetPlatform.iOS
            ? ResolutionPreset.low
            : ResolutionPreset.high,
        enableAudio: false);
    initializeControllerFuture = camera.initialize().then((value) {
      AppHelper.log(
          "_initializeCamera", "Camera initialized, starting camera
stream..");

      camera.startImageStream((CameraImage image) {
        if (!TFLiteHelper.modelLoaded) return;
        if (isDetecting) return;
        isDetecting = true;
        try {
          TFLiteHelper.classifyImage(image);
        } catch (e) {
          print(e);
        }
      });
    });
  }
}
```

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
GUIA DE PRACTICA DE LABORATORIO /TALLER

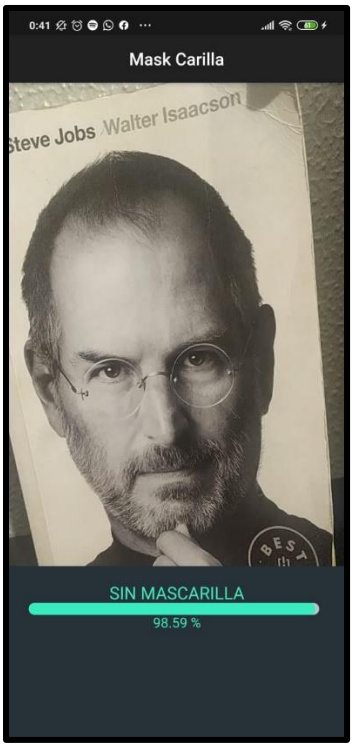
app_helper.dart

```dart
import 'package:flutter/cupertino.dart';

/// Clase para imprimrir mensajes en consola
class AppHelper {
  /// Metodo static que imprime mensajes en la consola
  static void log(String methodName, String message) {
    debugPrint("{$methodName} {$message}");
  }
}
```

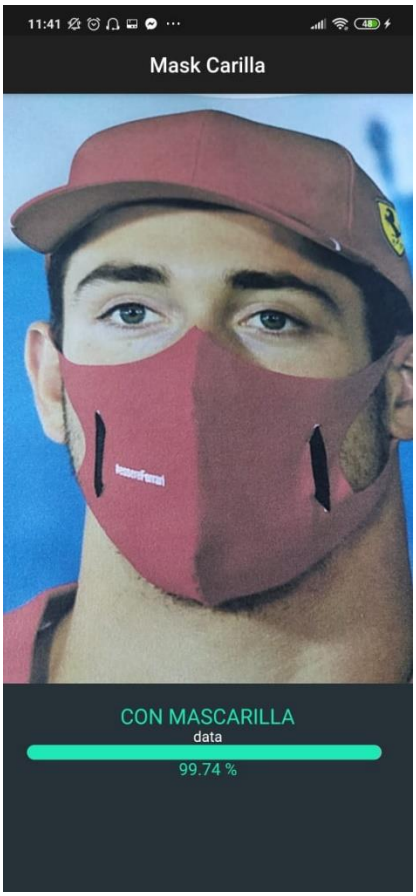
Pantallas

Detectando
Persona con mascarilla



DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
GUIA DE PRACTICA DE LABORATORIO /TALLER

Persona sin mascarilla



Conclusiones.

- Por medio de TeachableMachine y su opción de proyecto con imágenes permite entrenar un modelo en base a nuestros requerimientos, permite exportarlo para ser usado en Flutter a través de su dependencia de tflite.
- Se puede que el marco de desarrollo de Flutter permite crear aplicaciones para Android y iOS, con una solo porción de código y acoplándose a las restricciones de cada sistema operativo.
- Para crear una red neuronal se debe contar con al menos 1000 imágenes para cada caso y entrenarlo con 500 epochs,

Recomendaciones.

- Utilizar Flutter en VSCode porque en Android Studio consume muchos recursos.
- Antes de exportar el modelo entrenado se debe probarlo en el sitio web, en caso que exista falta de precisión se debe volver a entrenarlo

4. Bibliografía

Complete Guide to Artificial Neural Network Concepts & Models. (s. f.). MissingLink.ai. Recuperado 31 de agosto de 2020, de https://missinglink.ai/guides/neural-network-concepts/complete-guide-artificial-neural-networks/

Flutter. El nuevo framework de Google. (s. f.). Recuperado 31 de agosto de 2020, de https://www.itdo.com/blog/flutter-el-nuevo-framework-de-google/

Google actualiza Teachable Machine con una inteligencia artificial. (s. f.). Recuperado 31 de agosto de 2020, de https://www.fayerwayer.com/2019/11/teachable-machine-es-actualizada-con-una-inteligencia-artificial/

DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN
GUIA DE PRACTICA DE LABORATORIO /TALLER

Teachable Machine, la inteligencia artificial de Google, te responde con un GIF. (s. f.). Recuperado

31 de agosto de 2020, de https://nmas1.org/news/2017/10/09/inteligencia-artificial