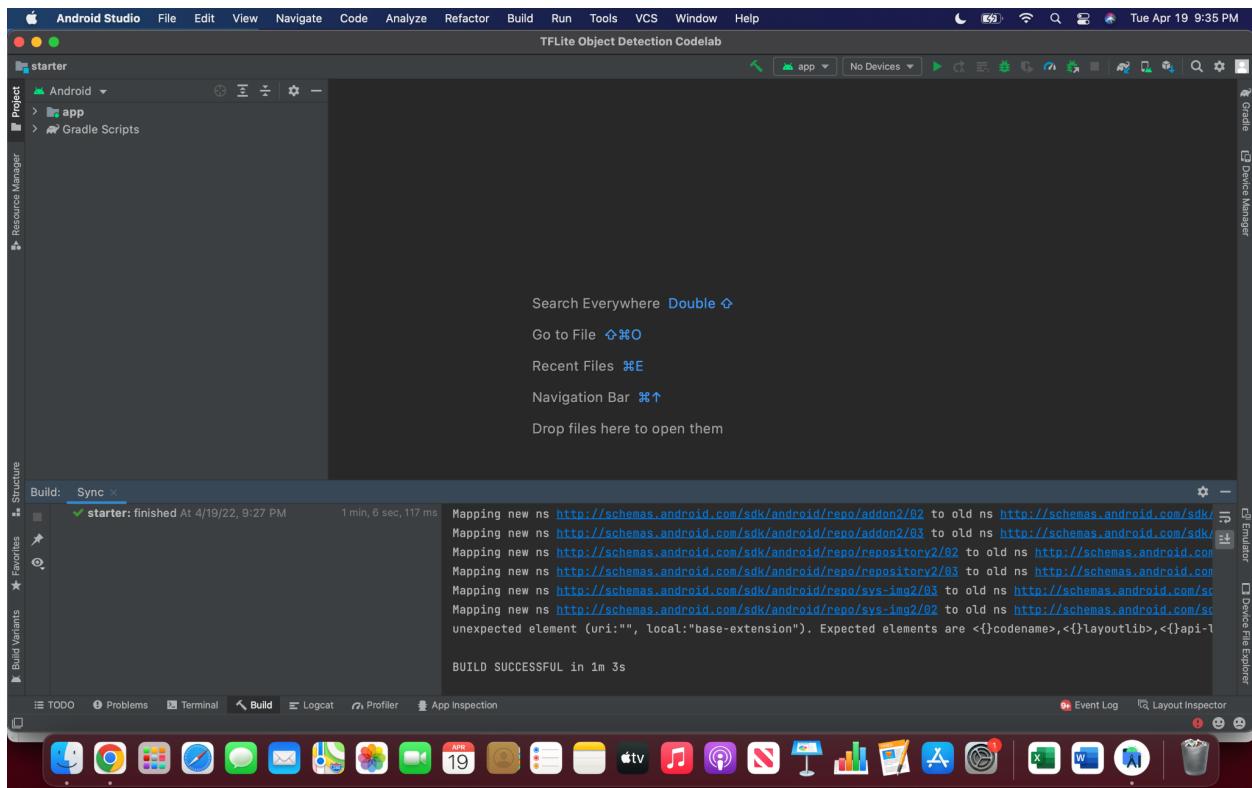


CMPE 258 – ASSIGNMENT 5 PART 2
Advanced Keras and Intro to edge ML

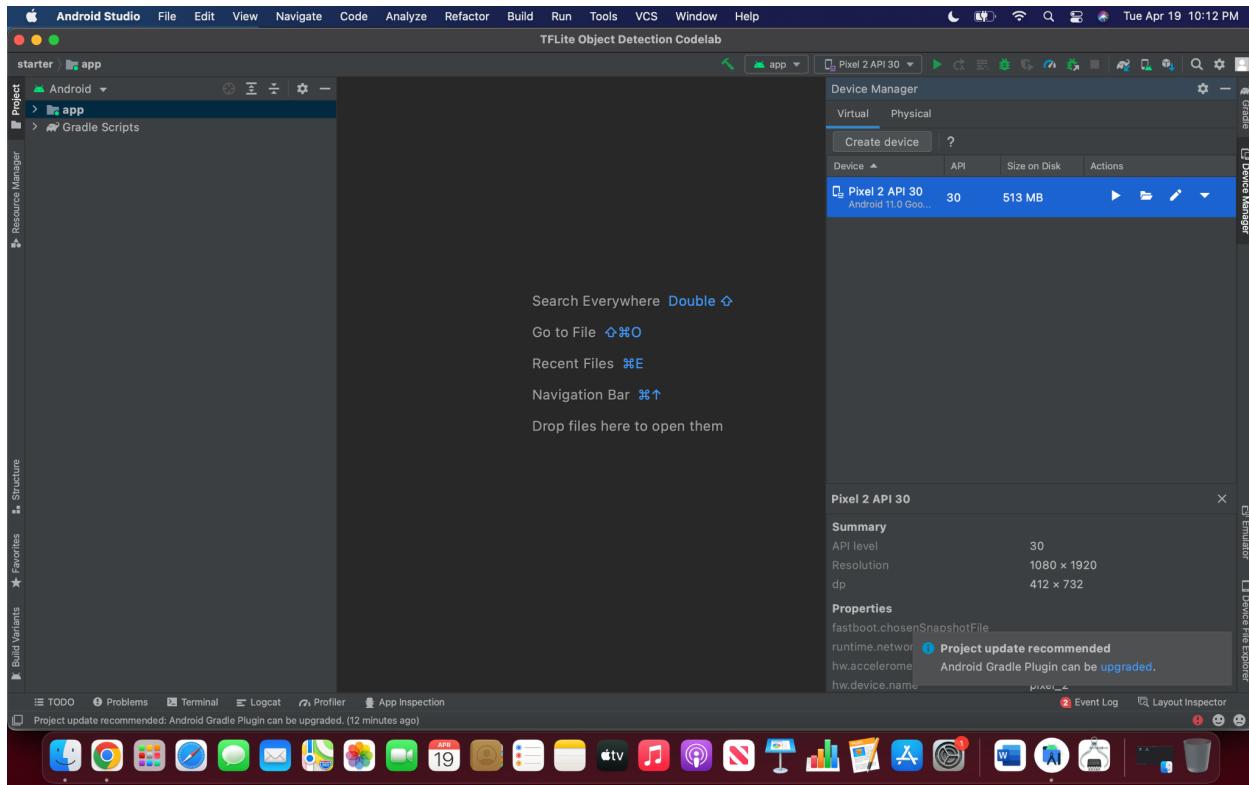
SUBMITTED BY: SWATHI ANANDRAM

A) Mobile App and Image training on device: Build and deploy a custom object detection model using tflite on android simulator

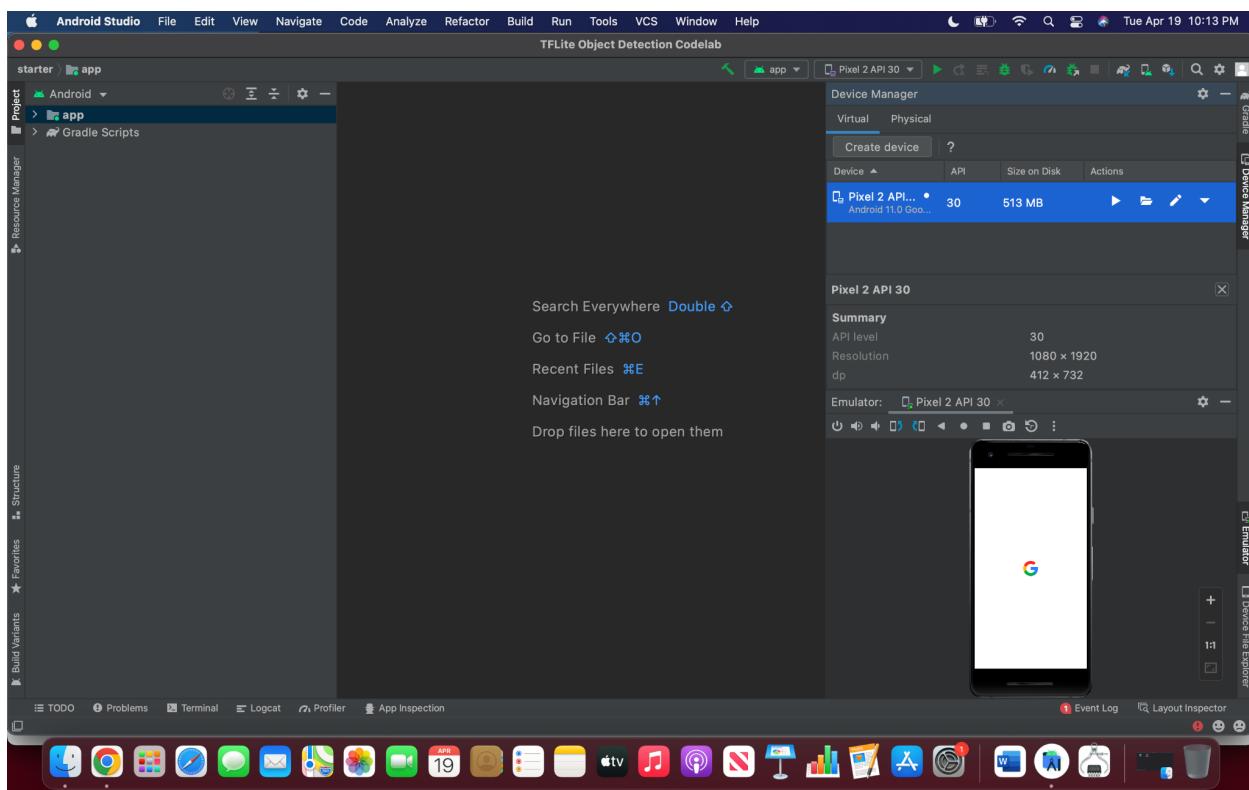
- Download Android Studio and Import Started Project



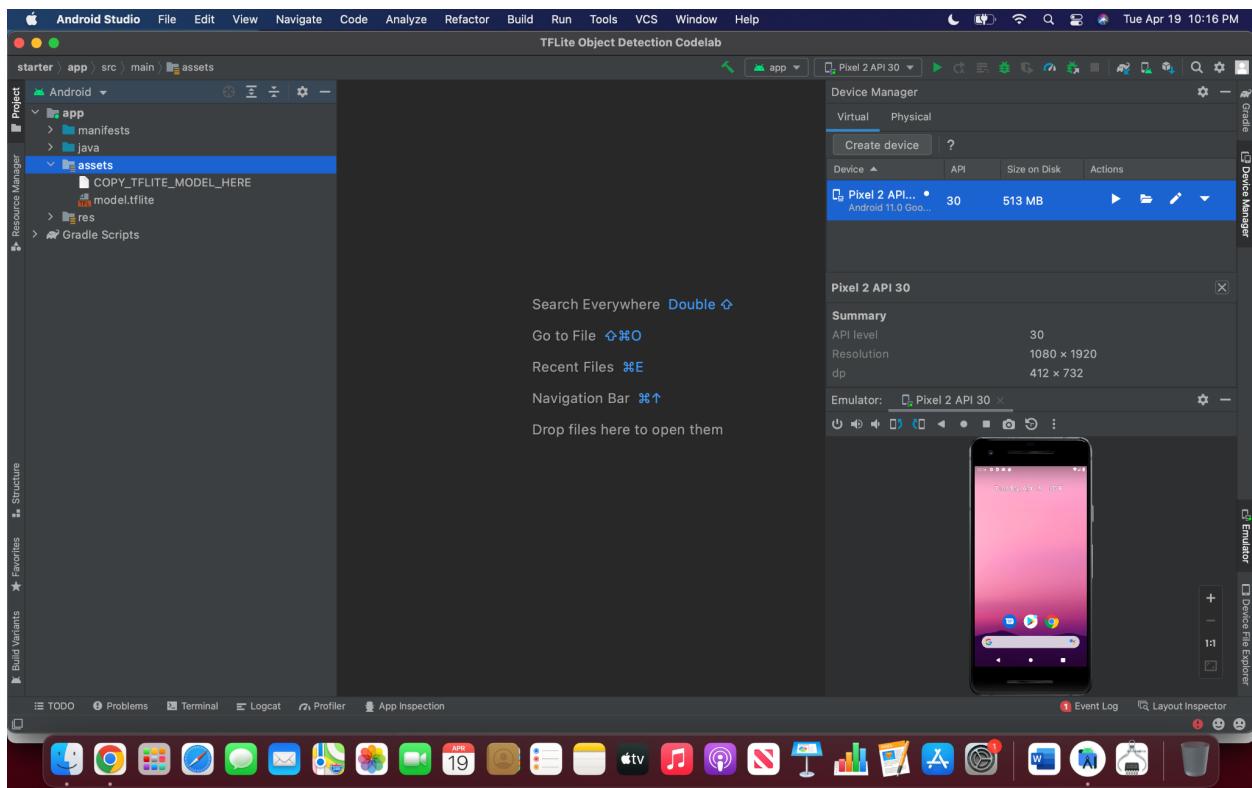
- Adding Virtual Device



- Run Virtual Device



- Adding the model to the starter project

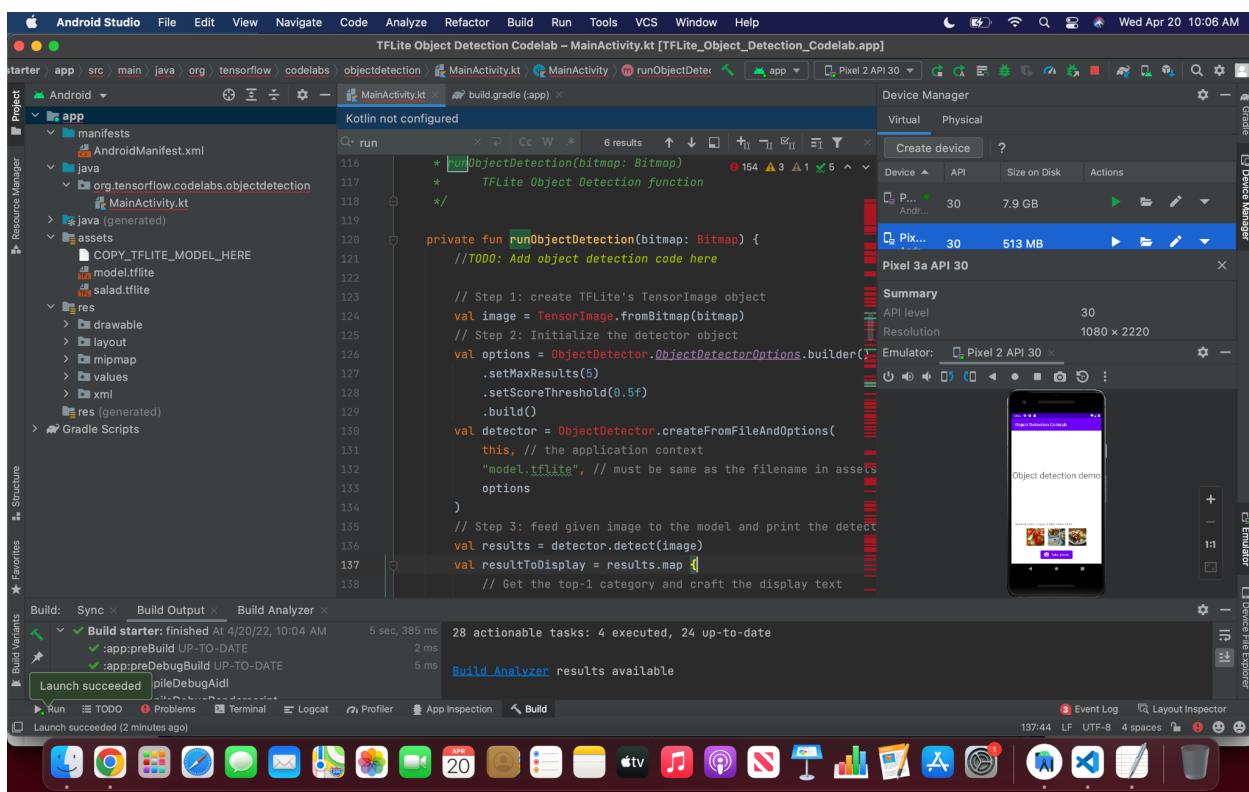
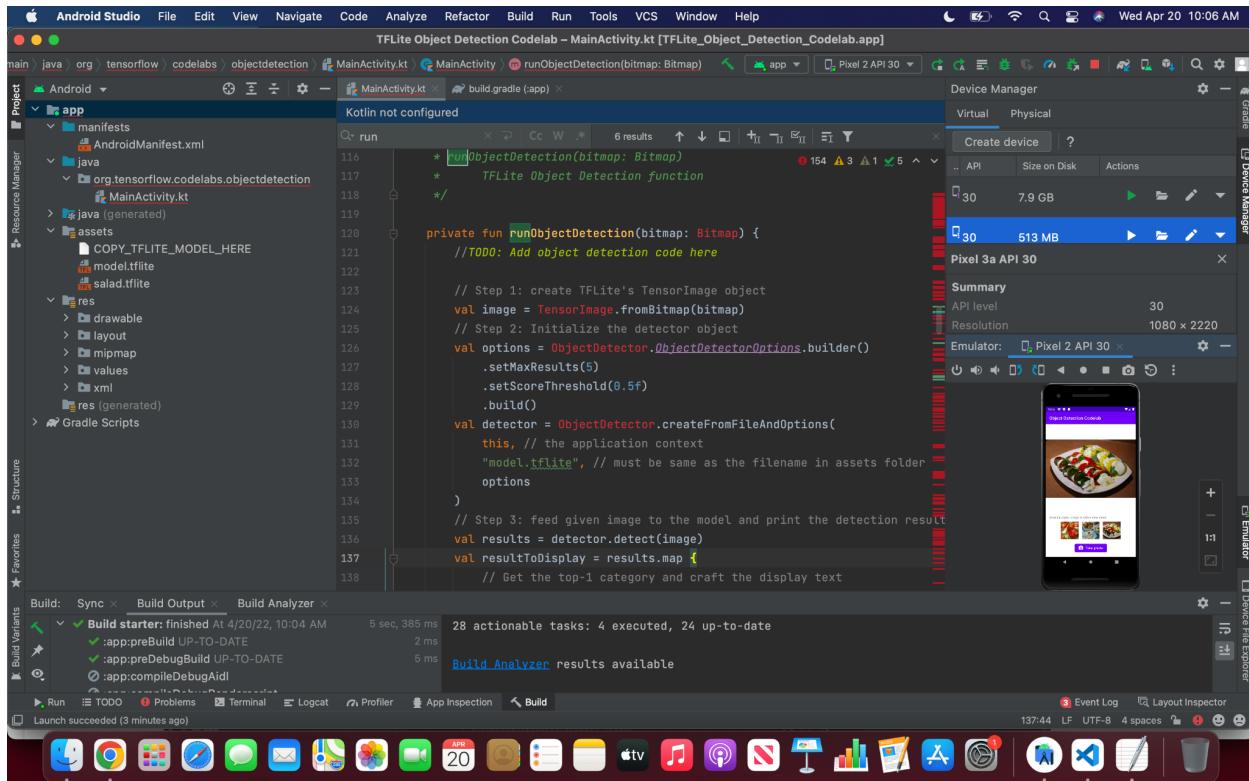


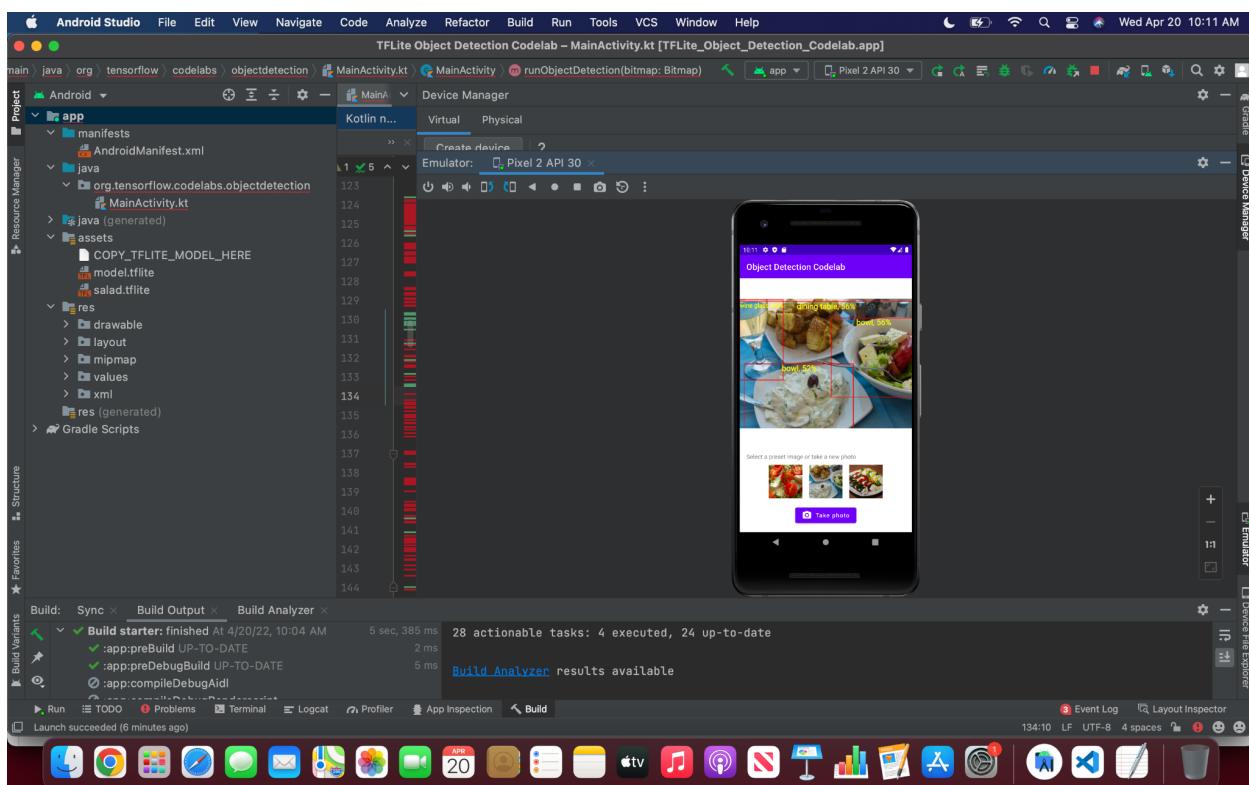
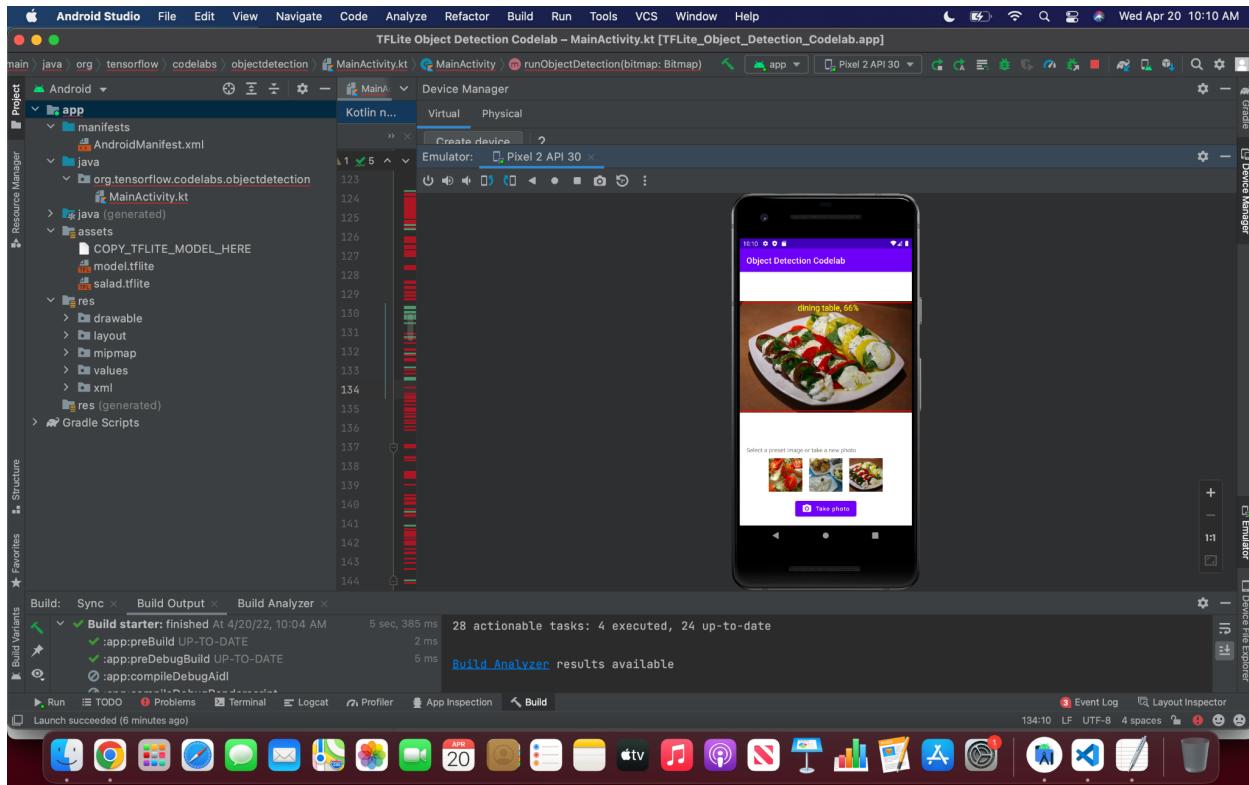
- Adding dependency in build.gradle

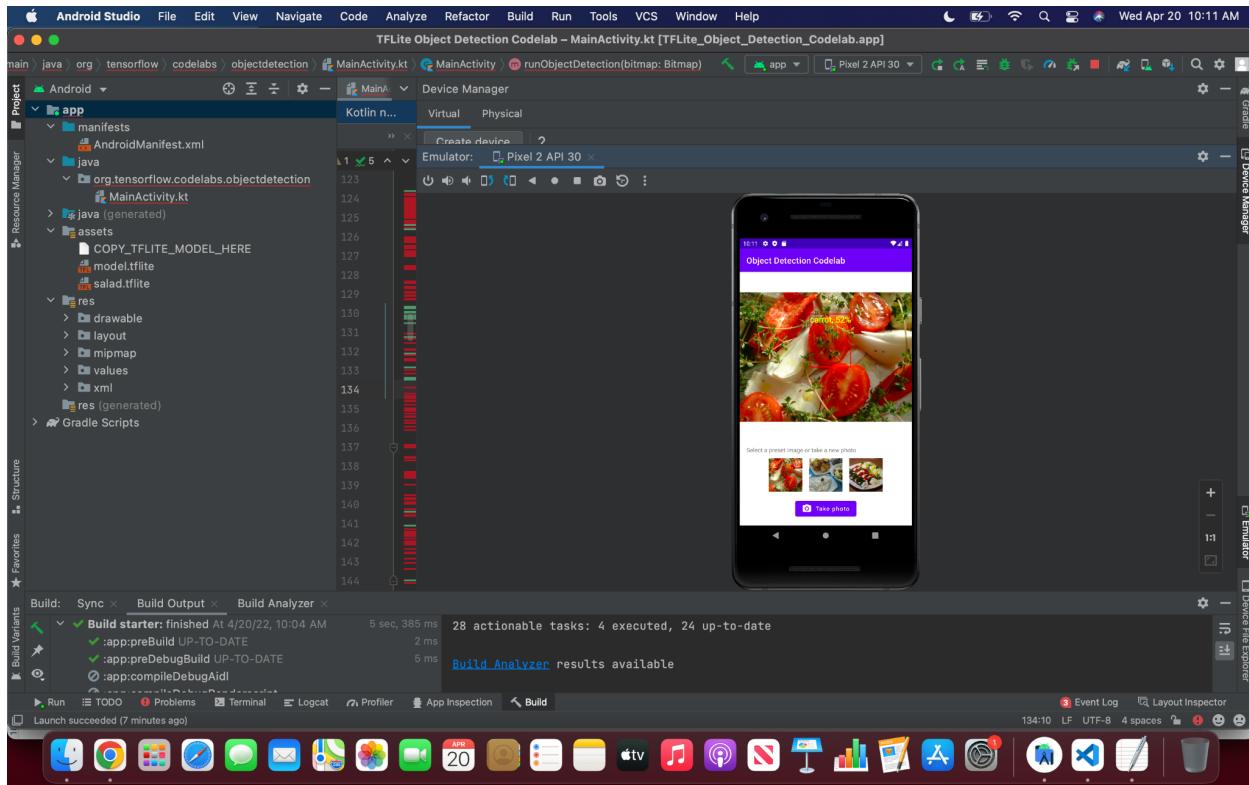
```
build.gradle
Users > swathananandram > Downloads > odml-pathways-main > object-detection > codelab2 > android > starter > app > build.gradle
27 |     sourceCompatibility JavaVersion.VERSION_1_8
28 |     targetCompatibility JavaVersion.VERSION_1_8
29 |
30 |     kotlinOptions {
31 |         jvmTarget = '1.8'
32 |     }
33 }
34
35 dependencies {
36
37     implementation "org.jetbrains.kotlin:kotlin-stdlib:$kotlin_version"
38     implementation 'androidx.core:core-ktx:1.3.2'
39     implementation 'androidx.appcompat:appcompat:1.2.0'
40     implementation 'com.google.android.material:material:1.3.0'
41     implementation 'androidx.constraintlayout:constraintlayout:2.0.4'
42     implementation 'org.jetbrains.kotlinx:kotlinx-coroutines-core:1.4.2'
43     implementation 'androidx.lifecycle:lifecycle-runtime-ktx:2.3.0'
44     implementation 'androidx.exifinterface:exifinterface:1.3.2'
45     implementation 'org.tensorflow:tensorflow-lite-task-vision:0.3.1'
46
47 //TODO: Add TFLite Task Library here
```

Ln 45, Col 70 Spaces: 4 UTF-8 LF Groovy ⌂

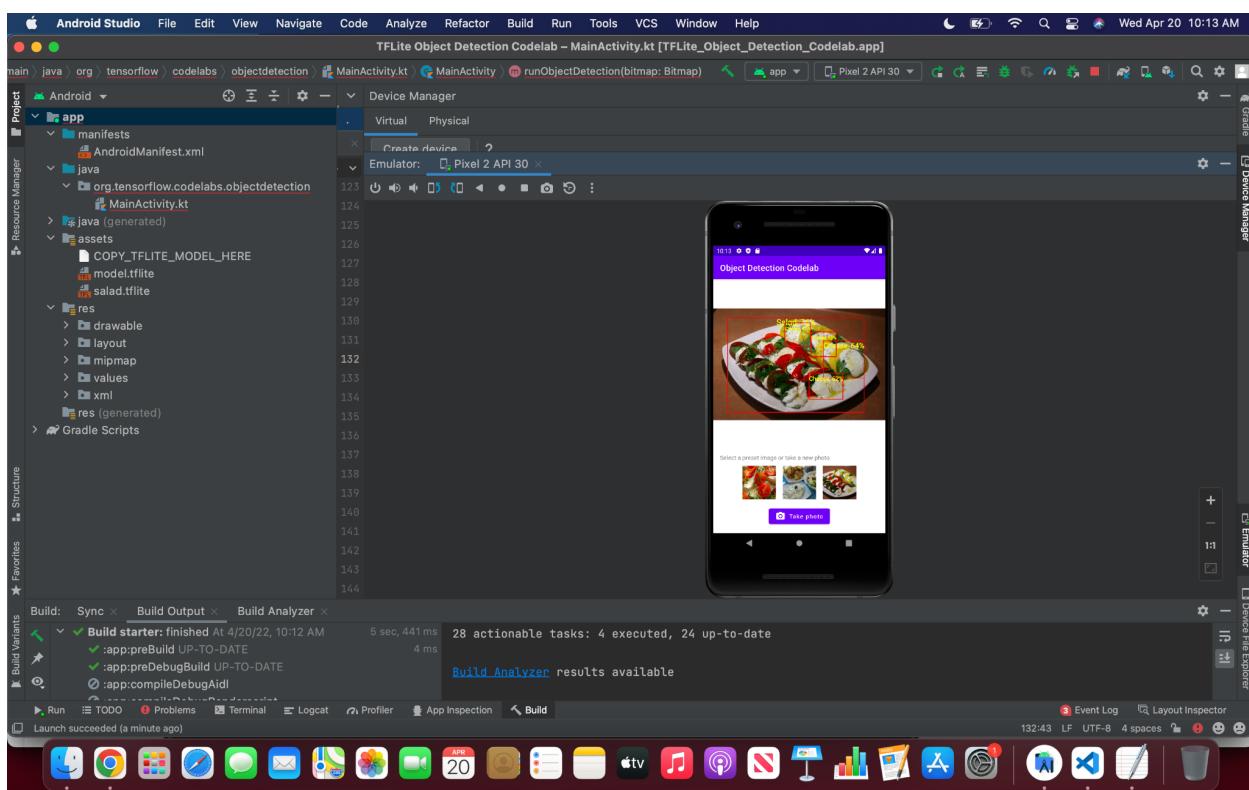
- Create Image Object, Create a Detector instance , Feed Image(s) to the detector



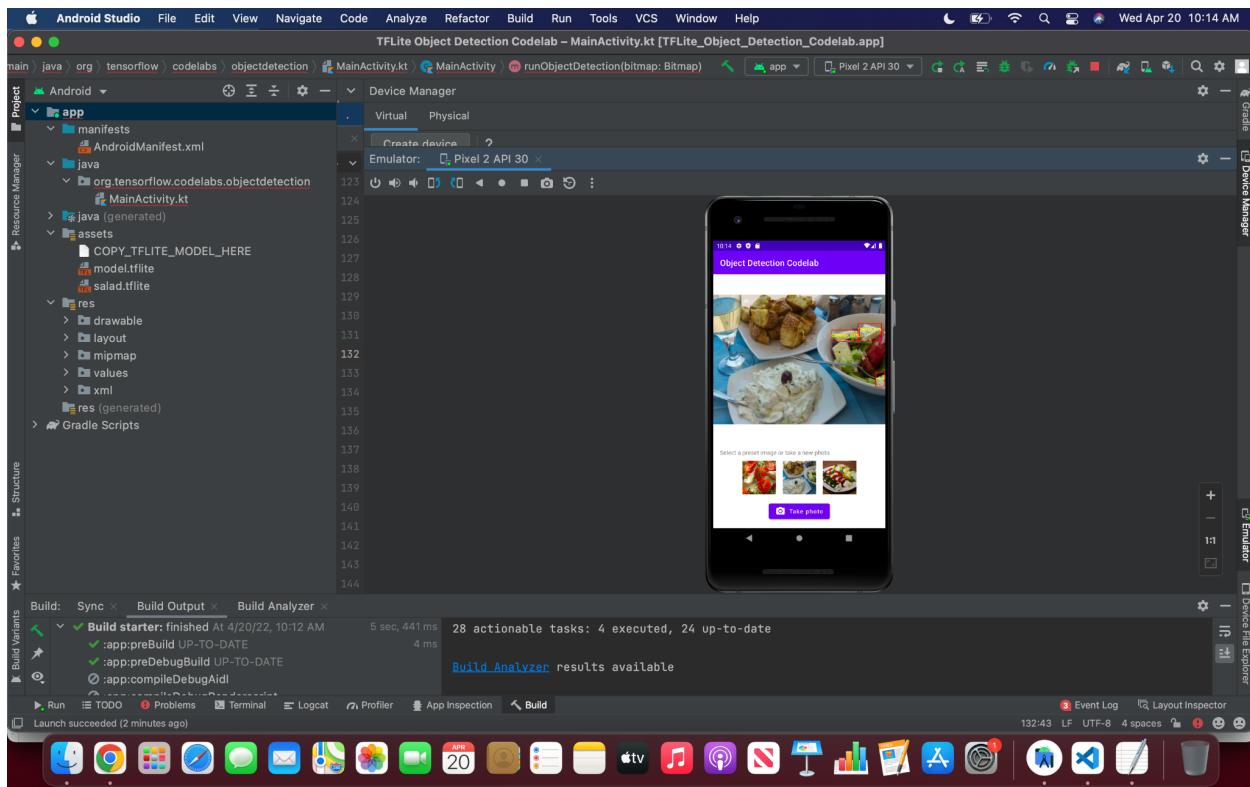
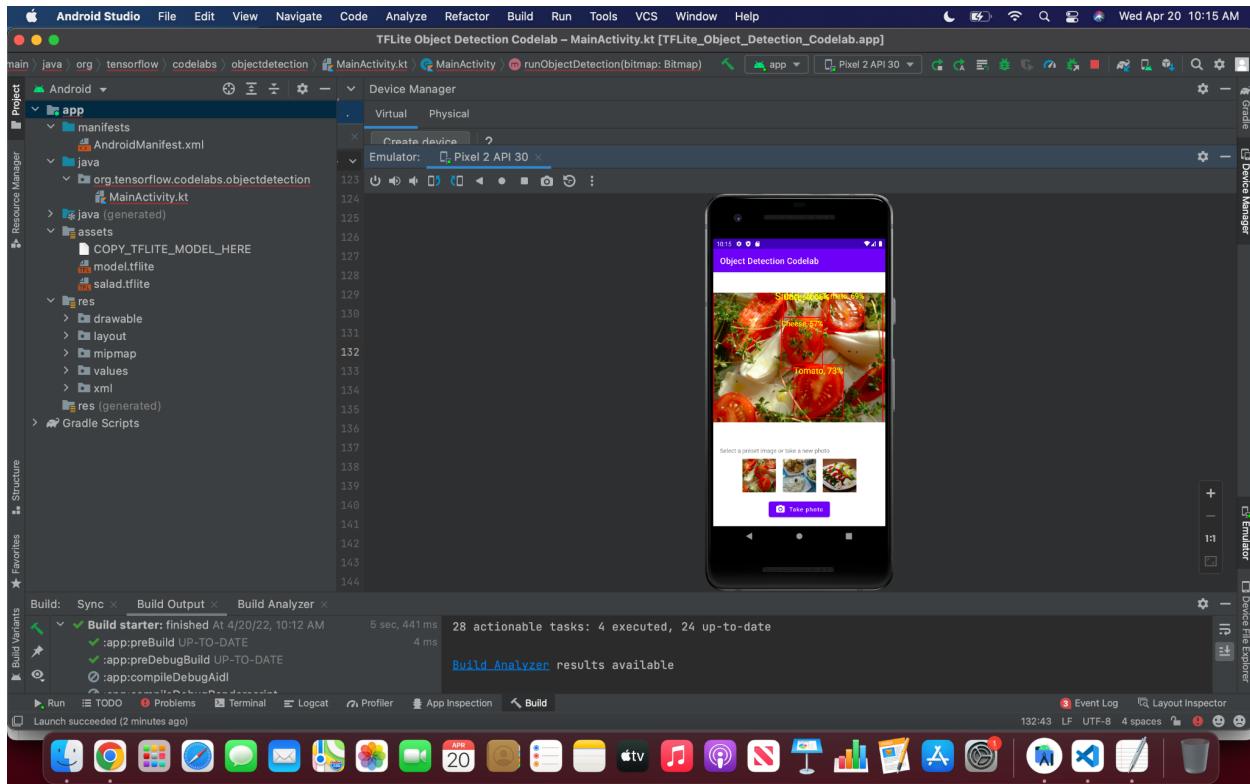




- Changing model

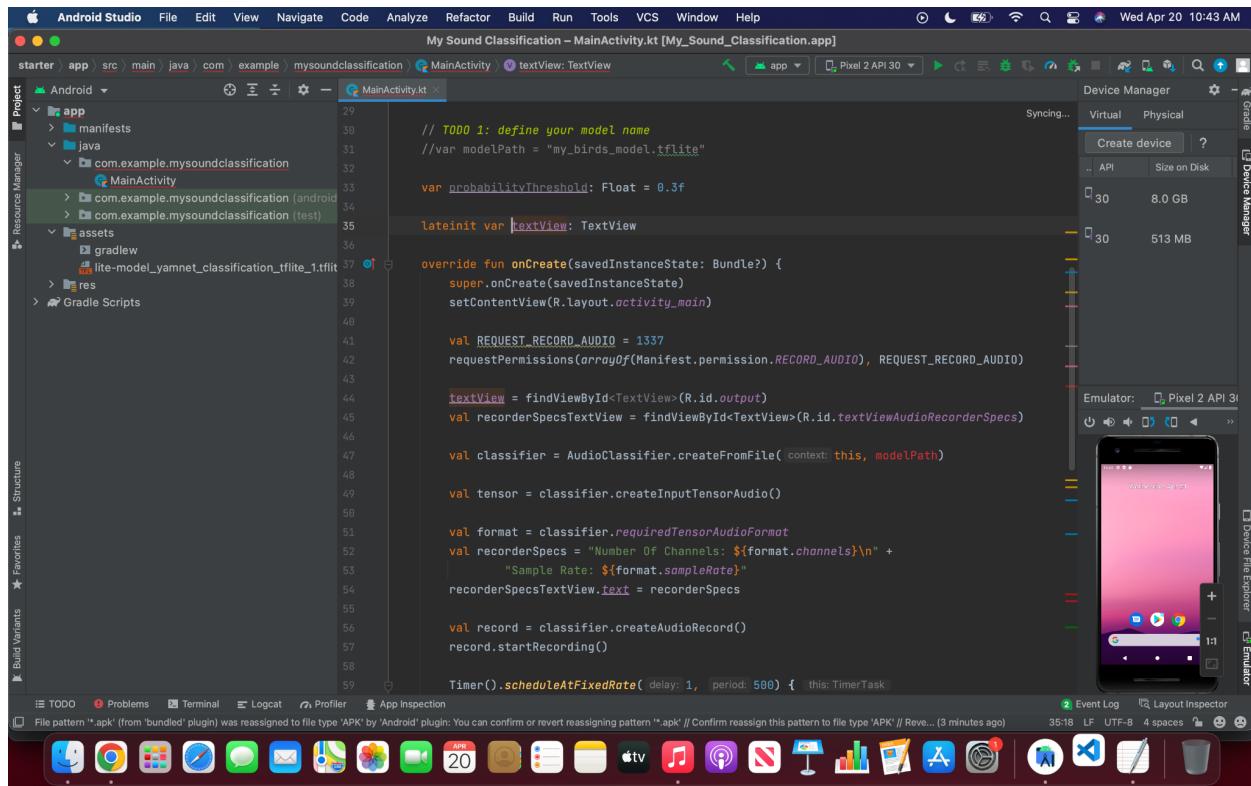


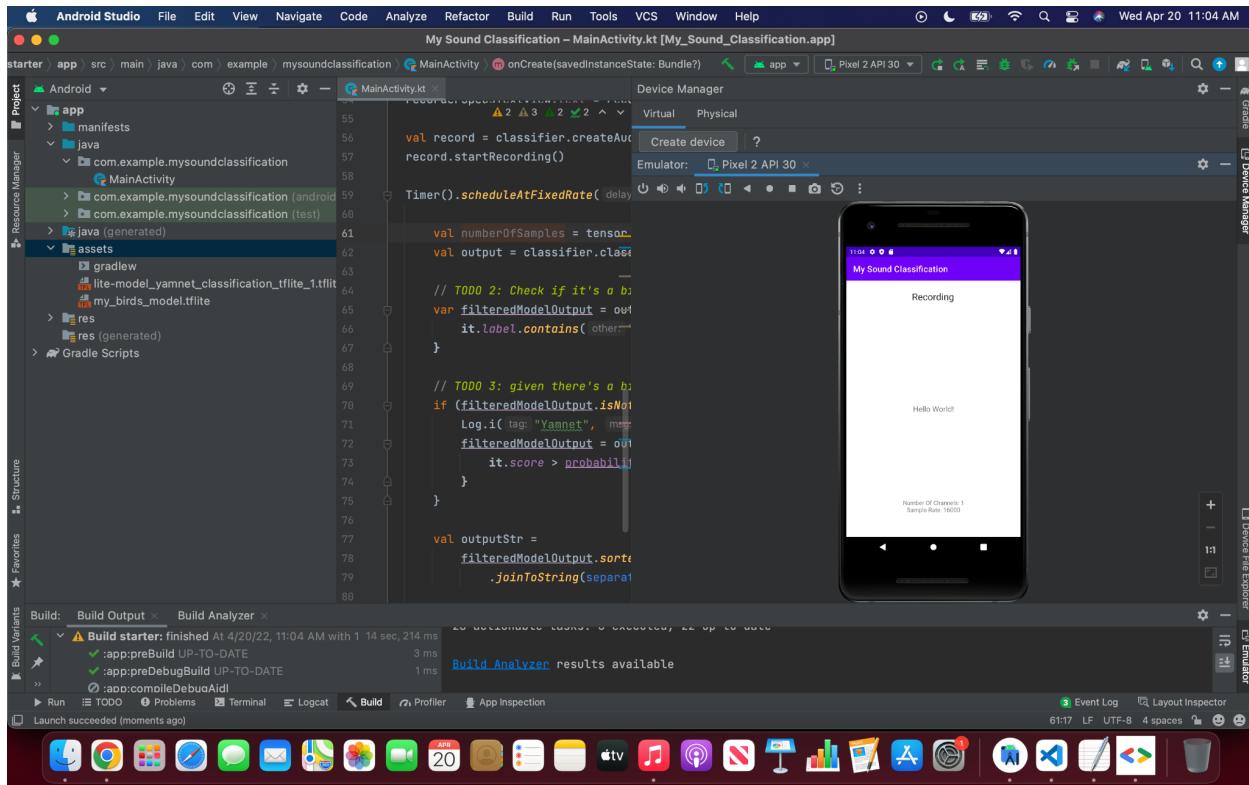
As we can see the bounding boxes are able to recognize various elements on the plate unlike the previous model which detected it as dining table



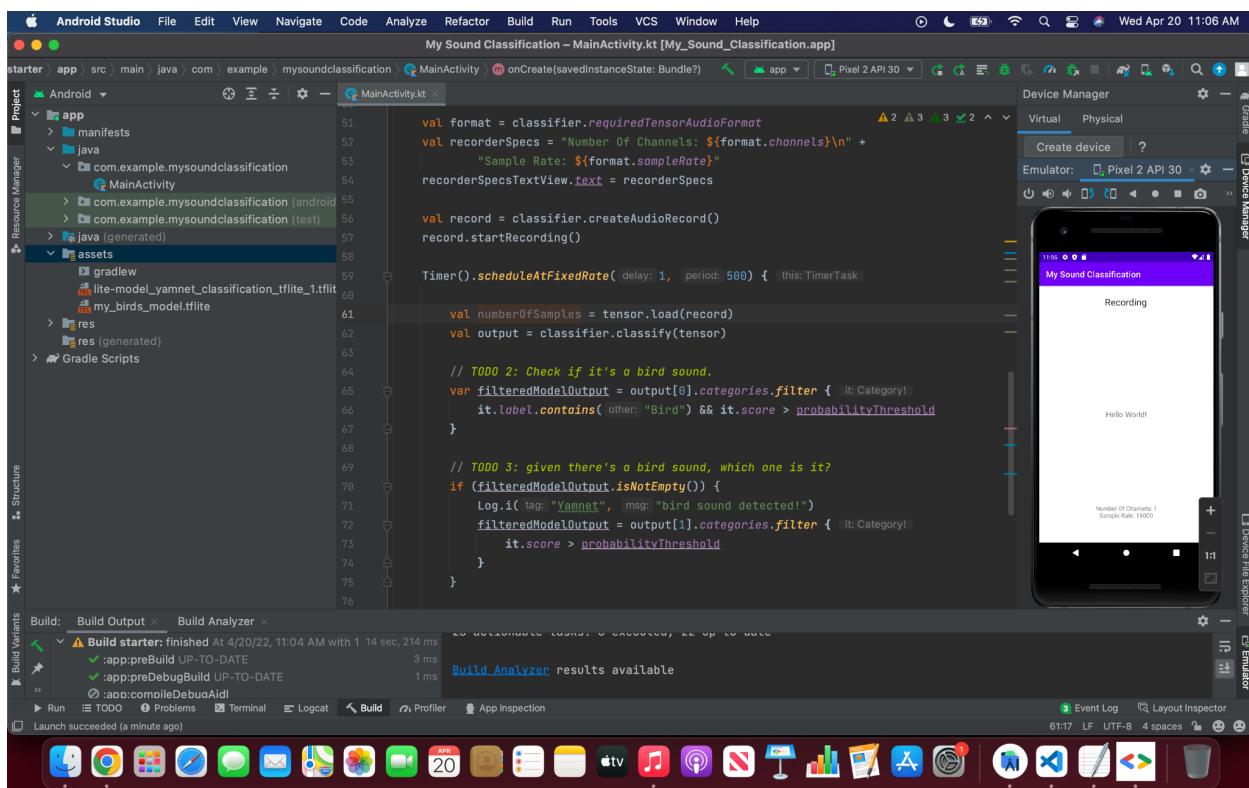
B) Mobile app and audio training on device : Build a custom pre-trained Audio Classification model

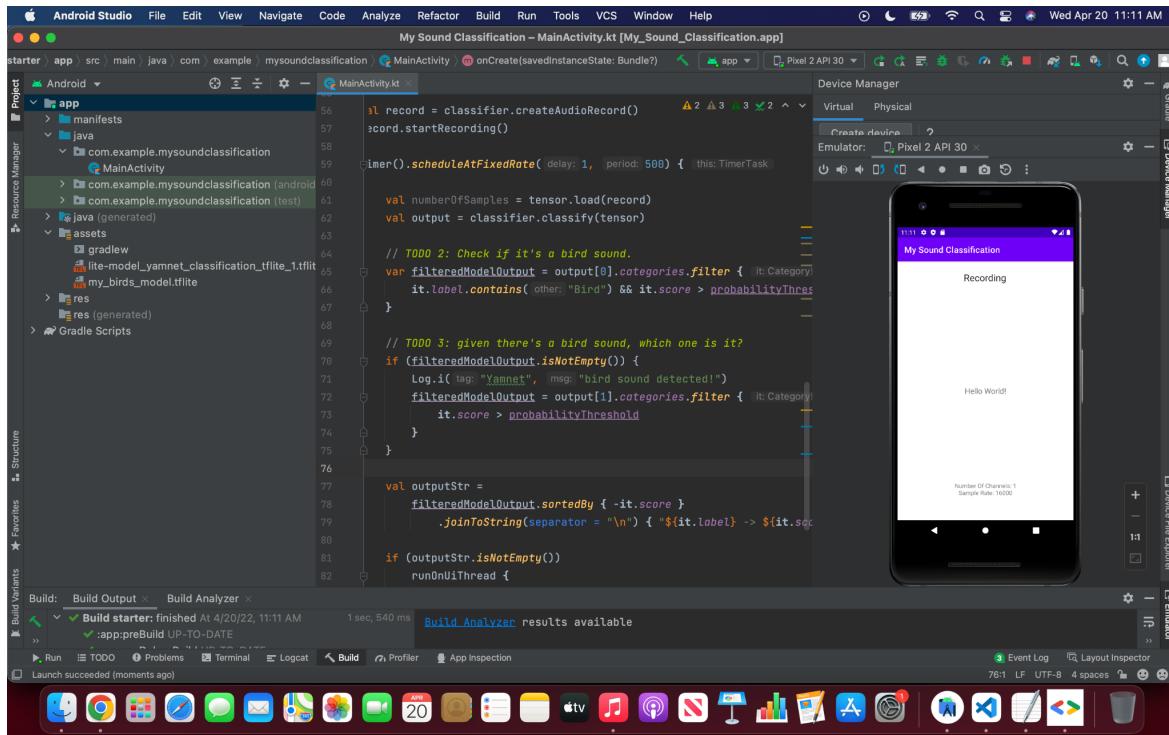
- Importing the starter project and starting the emulator, Loading the model, Creating the audio recorder and beginning the recording



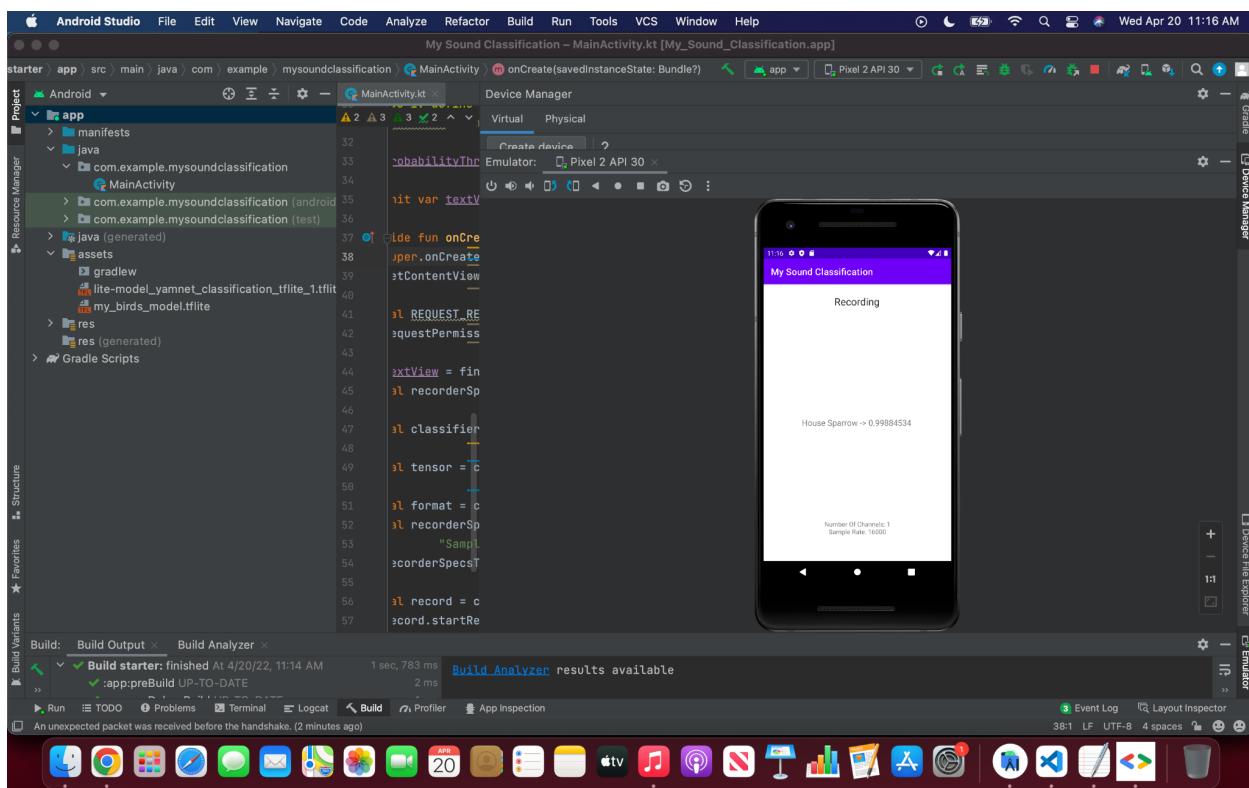


- Change the filtering to also filter out anything that is not Bird



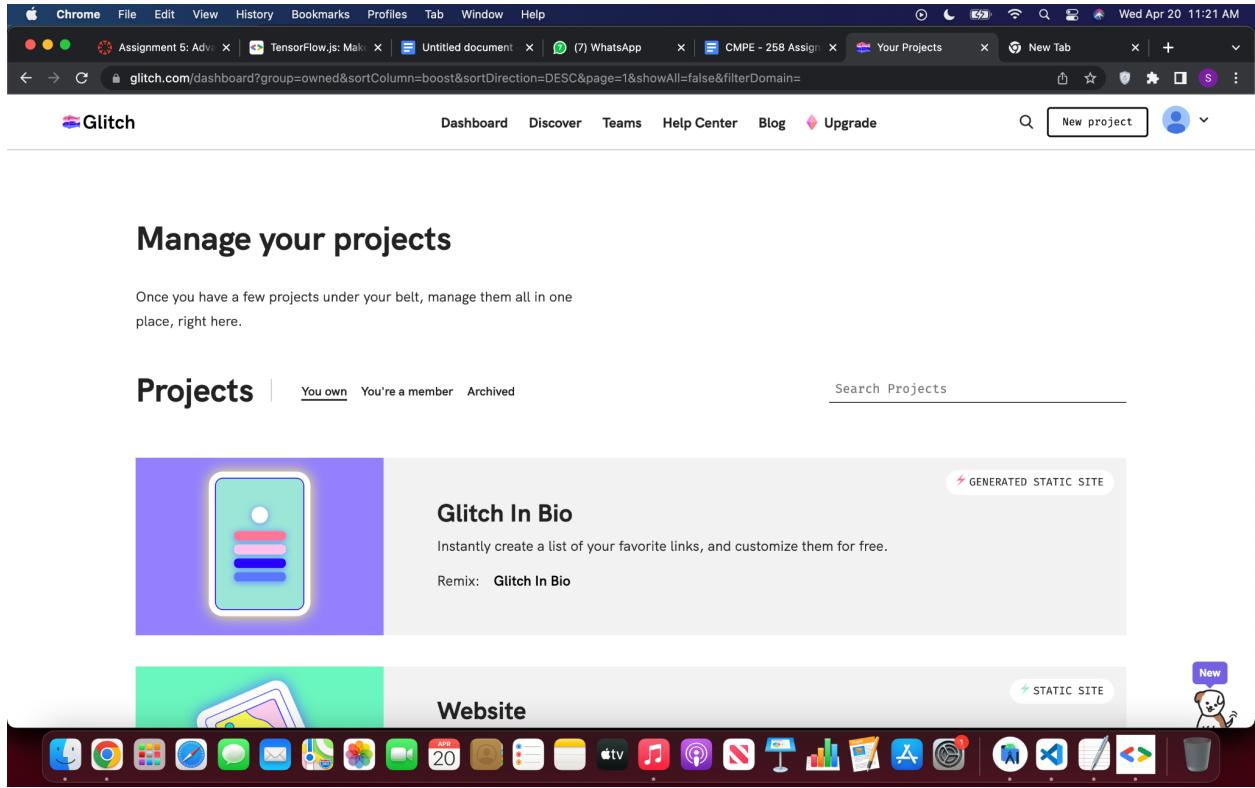


- Predicted Results when played a sparrow sound which means the model is working fine



C) Web app and image training on device: TensorFlow.js Transfer Learning Image Classifier

- Create a Glitch profile



- Get the boilerplate template

The hello world for TensorFlow.js :-) Absolute minimum needed to import into your website and simply prints the loaded TensorFlow.js version. From here we can do great things. Clone this to make your own TensorFlow.js powered projects or if you are following a tutorial that needs TensorFlow.js to work.

[Save](#) [Thank](#) [Share](#) [Add to playlist](#) [View source](#)

TensorFlow.js Boilerplate

This very simple skeleton simply loads in TensorFlow.js and prints out the version once loaded to the DOM.

From these humble beginnings you can do some really great things.

Feel free to fork this and use it as a quick way to start coding with TensorFlow.js directly or following some other tutorial that needs TensorFlow.js to run.

Your Project

[index.html](#)

We simply have a script tag in our HTML to grab the latest version of TensorFlow.js

In this case we simply reference the following:

```
<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs/dist/tf.min.js">
```

However, if you want to pull in a particular version of TensorFlow.js you can do so like this:

```
<script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@1.4.0/dist/tf.min.js">
```

Optional: If you want to include our TF.js visualization library you can do so using this

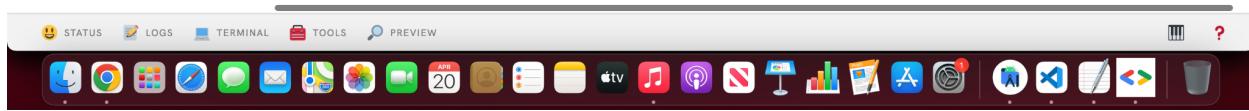
- APP HTML boilerplate changes

● **index.html**

```

<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Transfer Learning - TensorFlow.js</title>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- Import the webpage's stylesheet -->
    <link rel="stylesheet" href="/style.css">
  </head>
  <body>
    <h1>Make your own "Teachable Machine" using Transfer Learning with MobileNet v3 in TensorFlow.js using saved graph model from TFHub.</h1>
    <p id="status">Awaiting TF.js load</p>
    <video id="webcam" autoplay muted></video>
    <button id="enableCam">Enable Webcam</button>
    <button class="dataCollector" data-ihot="0" data-name="Class 1">Gather Class 1 Data</button>
    <button class="dataCollector" data-ihot="1" data-name="Class 2">Gather Class 2 Data</button>
    <button id="train">Train & Predict!</button>
    <button id="reset">Reset</button>
  </body>
</html>

```



● Add styles

● **style.css**

```

body {
  font-family: helvetica, arial, sans-serif;
  margin: 2em;
}

h1 {
  font-style: italic;
  color: #FF6F00;
}

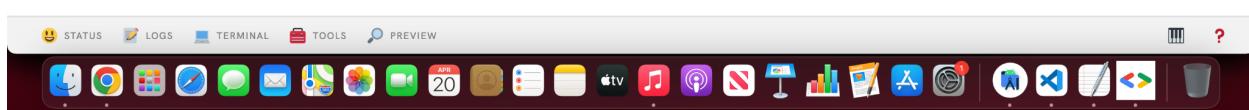
video {
  clear: both;
  display: block;
  margin: 10px;
  background: #000000;
  width: 640px;
  height: 480px;
}

button {
  padding: 10px;
  float: left;
  margin: 5px 3px 5px 10px;
}

.removed {
  display: none;
}

#status {
  font-size: 150%;
}

```



- Preview the changes

```

body {
    font-family: helvetica, arial, sans-serif;
    margin: 2em;
}
h1 {
    font-style: italic;
    color: #FF6700;
}
video {
    clear: both;
    display: block;
    margin: auto;
    background: #000000;
    width: 640px;
    height: 480px;
}
button {
    padding: 10px;
    float: left;
    margin: 5px 3px 5px 10px;
}
.removed {
    display: none;
}
#status {
    font-size: 150%;
}

```

- Define Key Constants

```

const STATUS = document.getElementById('status');
const VIDEO = document.getElementById('webcam');
const ENABLE_CAM_BUTTON = document.getElementById('enableCam');
const RESET_BUTTON = document.getElementById('reset');
const TRAIN_BUTTON = document.getElementById('train');
const MOBILE_NET_INPUT_WIDTH = 224;
const MOBILE_NET_INPUT_HEIGHT = 224;
const STOP_DATA_GATHER = -1;
const CLASS_NAMES = [];

```

- Add key event listeners

script.js

```

1 const STATUS = document.getElementById('status');
2 const VIDEO = document.getElementById('webcam');
3 const ENABLE_CAM_BUTTON = document.getElementById('enableCam');
4 const RESET_BUTTON = document.getElementById('reset');
5 const TRAIN_BUTTON = document.getElementById('train');
6 const MOBILE_NET_INPUT_WIDTH = 224;
7 const MOBILE_NET_INPUT_HEIGHT = 224;
8 const STOP_DATA_GATHER = -1;
9 const CLASS_NAMES = [];
10
11 ENABLE_CAM_BUTTON.addEventListener('click', enableCam);
12 TRAIN_BUTTON.addEventListener('click', trainAndPredict);
13 RESET_BUTTON.addEventListener('click', reset);
14
15 function enableCam() {
16   // TODO: Fill this out later in the codelab!
17 }
18
19
20 function trainAndPredict() {
21   // TODO: Fill this out later in the codelab!
22 }
23
24
25 function reset() {
26   // TODO: Fill this out later in the codelab!
27 }
28 
```

Make your own "Teachable Machine" using Transfer Learning with MobileNet v3 in TensorFlow.js using saved graph model from TFHub.

Awaiting TF.js load

- you can find all buttons that have a class of 'dataCollector' using `document.querySelectorAll()`

The screenshot shows a Mac desktop with a Glitch project open in a browser window. The project is titled "script.js" and contains the following code:

```

20
21  function trainAndPredict() {
22    // TODO: Fill this out later in the codelab!
23  }
24
25
26  function reset() {
27    // TODO: Fill this out later in the codelab!
28  }
29
30  let dataCollectorButtons = document.querySelectorAll('button.dataCollector');
31  for (let i = 0; i < dataCollectorButtons.length; i++) {
32    dataCollectorButtons[i].addEventListener('mousedown', gatherDataForClass);
33    dataCollectorButtons[i].addEventListener('mouseup', gatherDataForClass);
34    // Populate the human readable names for classes.
35    CLASS_NAMES.push(dataCollectorButtons[i].getAttribute('data-name'));
36  }
37
38
39  function gatherDataForClass() {
40    // TODO: Fill this out later in the codelab!
41  }

```

The right side of the screen displays the URL "majestic-ambiguous-spaghetti.glitch.me/" and a large orange text overlay:

Make your own "Teachable Machine" using Transfer Learning with MobileNet v3 in TensorFlow.js using saved graph model from TFHub.

Below the URL, it says "Awaiting TF.js load" and shows a large black rectangular placeholder.

At the bottom of the screen is the Mac OS X dock with various application icons.

- add some variables to store key things

The screenshot shows the same Glitch project as before, but now with additional variables added to the "script.js" file:

```

28
29
30  let dataCollectorButtons = document.querySelectorAll('button.dataCollector');
31  for (let i = 0; i < dataCollectorButtons.length; i++) {
32    dataCollectorButtons[i].addEventListener('mousedown', gatherDataForClass);
33    dataCollectorButtons[i].addEventListener('mouseup', gatherDataForClass);
34    // Populate the human readable names for classes.
35    CLASS_NAMES.push(dataCollectorButtons[i].getAttribute('data-name'));
36  }
37
38
39  function gatherDataForClass() {
40    // TODO: Fill this out later in the codelab!
41  }
42
43  let mobilenet = undefined;
44  let gatherDataState = STOP_DATA_GATHER;
45  let videoPlaying = false;
46  let trainingDataInputs = [];
47  let trainingDataOutputs = [];
48  let examplesCount = [];
49  let predict = false;

```

The right side of the screen displays the URL "majestic-ambiguous-spaghetti.glitch.me/" and a large orange text overlay:

Make your own "Teachable Machine" using Transfer Learning with MobileNet v3 in TensorFlow.js using saved graph model from TFHub.

Below the URL, it says "Awaiting TF.js load" and shows a large black rectangular placeholder.

At the bottom of the screen is the Mac OS X dock with various application icons.

- define a new function called `loadMobileNetFeatureModel`

The screenshot shows a Glitch project titled "majestic-ambiguous-spaghetti". The project structure includes files like LICENSE.md, README.md, index.html, script.js, and style.css. The script.js file contains code for loading a MobileNet v3 model from TFHub. A status message "MobileNet v3 loaded successfully!" is displayed in the preview window. The preview window also features buttons for "Enable Webcam", "Gather Class 1 Data", "Gather Class 2 Data", and "Train & Predict!". The OS X dock at the bottom shows various application icons.

```

// Populates the human readable names for classes.
CLASS_NAMES.push(dataCollectorButtons[i].getAttribute("data-name"));
}

function gatherDataForClass() {
  // TODO: Fill this out later in the codelab!
}

let mobilenet = undefined;
let gatherDataState = STOP_DATA_GATHER;
let videoPlaying = false;
let trainingDataInputs = [];
let trainingDataOutputs = [];
let completedCount = 0;
let predict = false;

/**
 * Loads the MobileNet model and warms it up so ready for use.
 */
async function loadMobileNetFeatureModel() {
  const URL = "https://tfhub.dev/google/tfjs-model/imagenet/mobilenet_v3_small_100_224/feature_vector/5/default/1";
  mobilenet = await tf.loadGraphModel(URL, {fromTFHub: true});
  STATUS.innerHTML = "MobileNet v3 loaded successfully!";
}

// Warm up the model by passing zeros through it once.
tf.tidy(function () {
  let answer = mobilenet.predict(tf.zeros([1, MOBILE_NET_INPUT_HEIGHT, MOBILE_NET_INPUT_WIDTH, 3]));
  console.log(answer.shape);
});

// Call the function immediately to start loading.
loadMobileNetFeatureModel();
}

```

- status text change from "Awaiting TF.js load" to become "MobileNet v3 loaded successfully!"

Glitch

script.js

```
// Glitch
script.js PRETTIER
m majestic-ambiguous-spaghetti
Settings Assets Files LICENSE.md README.md index.html script.js style.css
51 // Populate the human readable names for classes.
52 CLASS_NAMES.push(dataCollectorButtons[i].getAttribute('name'));
53 }
54 }
55
56 function gatherDataForClass() {
57 // TODO: Fill this out later in the codable!
58 }
59
60 let mobilenet = undefined;
61 let gatherDataState = STOP_DATA_GATHER;
62 let videoPlaying = false;
63 let trainingDataInputs = [];
64 let trainingDataOutputs = [];
65 let recordedCount = 0;
66 let predict = false;
67
68 /**
69 * Loads the MobileNet model and warms it up so ready for
70 */
71 async function loadMobileNetFeatureModel() {
72 const URL = 'https://tfhub.dev/google/tfjs-model/imagenet/mobile';
73
74 mobilenet = await tf.loadGraphModel(URL, {fromTFHub: true});
75 STATUS.innerHTML = 'MobileNet v3 loaded successfully!';
76
77 // Warm up the model by passing zeros through it once.
78 tf.tidy(function () {
79 const answer = mobilenet.predict(tf.zeros([1, MOBILENET_V3_INPUT_SIZE]));
80 console.log(answer.shape);
81 });
82 }
83
84
85 // Call the function immediately to start loading.
86 loadMobileNetFeatureModel();
87
88
89 
```

Make your own "Teachable Machine" using Transfer Learning with MobileNet v3 in TensorFlow.js using saved graph model from TFHub.

MobileNet v3 loaded successfully!

Enable Webcam Gather Class 1 Data Gather Class 2 Data Train & Predict! Reset

- Define new Model head

Glitch

script.js PRETTIER

m majestic-ambiguous-spaghetti

Settings Assets Files LICENSE.md README.md index.html script.js style.css

```
let model = tf.sequential();
model.add(tf.layers.dense({inputShape: [1024], units: 128, activation: 'relu'});
model.add(tf.layers.dense({units: CLASS_NAMES.length, activation: 'softmax'}));
model.summary();
model.compile({
  optimizer: 'adam',
  loss: (CLASS_NAMES.length === 2) ? 'binaryCrossentropy' : 'categoricalCrossentropy',
  metrics: ['accuracy']
});
```

Make your own "Teachable Machine" using Transfer Learning with MobileNet v3 in TensorFlow.js using saved graph model from TFHub.

Awaiting TF.js load

STATUS LOGS TERMINAL TOOLS PREVIEW

- Enable the webcam

The screenshot shows a Mac desktop with a Glitch project open in a browser window. The project is titled 'Assignment 5: Advanced' and has a sub-project named 'TensorFlow.js: Make your own "Teachable Machine" using Transfer Learning with MobileNet v3 in TensorFlow.js using saved graph model from TFHub.'. The code in the script.js file is as follows:

```

1.09
131+ function hasGetUserMedia() {
132  return !(navigator.mediaDevices || navigator.mediaDevices.getUserMedia);
133 }
134+
135+ function enableCam() {
136  if (hasGetUserMedia()) {
137    // getUserMedia parameters.
138    const constraints = {
139      video: true,
140      width: 640,
141      height: 480
142    };
143+
144    // Activate the webcam stream.
145    navigator.mediaDevices.getUserMedia(constraints).then(function(stream) {
146      VIDEO.srcObject = stream;
147      VIDEO.addEventListener('loadeddata', function() {
148        videoPlaying = true;
149        ENABLE_CAM_BUTTON.classList.add('removed');
150      });
151    });
152  } else {
153    console.warn('getUserMedia() is not supported by your browser');
154  }
155}

```

The desktop also features a Dock at the bottom with various application icons.

- Checking the preview of webcam

The screenshot shows a Mac desktop with a Glitch project open in a browser window. The project is titled 'Assignment 5: Advanced' and has a sub-project named 'TensorFlow.js: Make your own "Teachable Machine" using Transfer Learning with MobileNet v3 in TensorFlow.js using saved graph model from TFHub.'. The code in the script.js file is as follows:

```

97  // As this is a classification problem you can record metrics: ['accuracy']
98  //}
99+
100+ function hasGetUserMedia() {
101  return !(navigator.mediaDevices || navigator.mediaDevices.getUserMedia);
102}
103+
104+ function enableCam() {
105  if (hasGetUserMedia()) {
106    // getUserMedia parameters.
107    const constraints = {
108      video: true,
109      width: 640,
110      height: 480
111    };
112+
113    // Activate the webcam stream.
114    navigator.mediaDevices.getUserMedia(constraints);
115    VIDEO.srcObject = stream;
116    VIDEO.addEventListener('loadeddata', function() {
117      videoPlaying = true;
118      ENABLE_CAM_BUTTON.classList.add('removed');
119    });
120  } else {
121    console.warn('getUserMedia() is not supported by your browser');
122  }
123}
124

```

A video preview window on the right shows a person's face. Below the video are buttons: 'Gather Class 1 Data', 'Gather Class 2 Data', 'Train & Predict!', and 'Reset'. The desktop also features a Dock at the bottom with various application icons.

- Data Collection button event handler

Glitch script.js PRETTIER

```

majestic-ambiguous-spaghetti <> script.js
  107   height: 480
  108 }
  109 // Activate the webcam stream.
  110 navigator.mediaDevices.getUserMedia(constraints).then(function(stream) {
  111   VIDEO.srcObject = stream;
  112   VIDEO.addEventListener('loadeddata', function() {
  113     videoPlaying = true;
  114     ENABLE_CAM_BUTTON.classList.add('removed');
  115   });
  116 } else {
  117   console.warn('getUserMedia() is not supported by your browser');
  118 }
  119 /**
  120  * Handle Data Gather for button mouseup/mousedown.
  121 */
  122 /**
  123  */
  124 function gatherDataForClass() {
  125   let classNumber = parseInt(this.getAttribute('data-1hot'));
  126   gatherDataState = (gatherDataState === STOP_DATA_GATHER) ? classNumber : STOP_DATA_GATHER;
  127   dataGatherLoop();
  128 }
  129 
```

Make your own "Teachable Machine" using Transfer Learning with MobileNet v3 in TensorFlow.js using saved graph model from TFHub.

MobileNet v3 loaded successfully!

- Data Collection

Glitch script.js PRETTIER

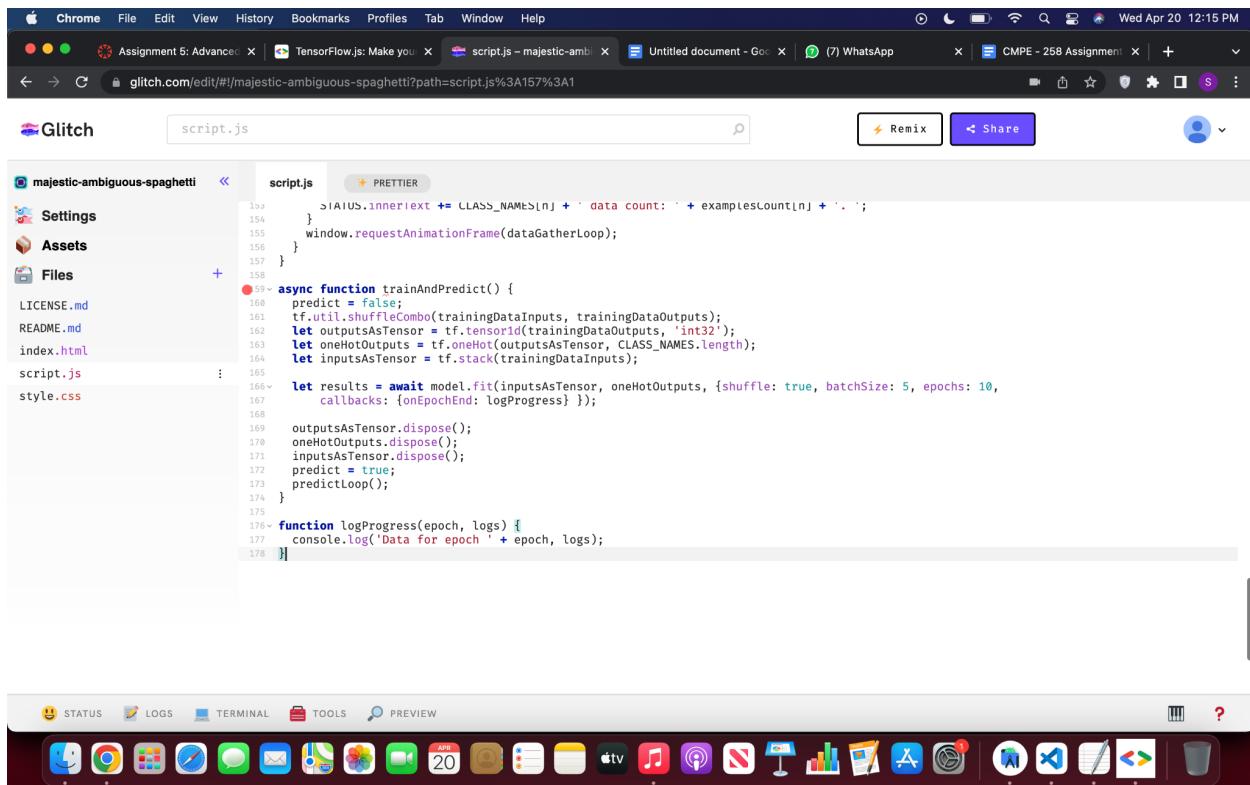
```

majestic-ambiguous-spaghetti <> script.js
  129   dataGatherLoop();
  130 }
  131 function dataGatherLoop() {
  132   if (videoPlaying && gatherDataState !== STOP_DATA_GATHER) {
  133     let imageFeatures = tf.tidy(function() {
  134       let videoFrameAsTensor = tf.browser.fromPixels(VIDEO);
  135       let resizedTensorFrame = tf.image.resizeBilinear(videoFrameAsTensor, [MOBILE_NET_INPUT_HEIGHT,
  136         MOBILE_NET_INPUT_WIDTH], true);
  137       let normalizedTensorFrame = resizedTensorFrame.div(255);
  138       return mobilenet.predict(normalizedTensorFrame.expandDims()).squeeze();
  139     });
  140     trainingDataInputs.push(imageFeatures);
  141     trainingDataOutputs.push(gatherDataState);
  142   }
  143   // Initialize array index element if currently undefined.
  144   if (examplesCount[gatherDataState] === undefined) {
  145     examplesCount[gatherDataState] = 0;
  146   }
  147   examplesCount[gatherDataState]++;
  148
  149   STATUS.innerText = '';
  150   for (let n = 0; n < CLASS_NAMES.length; n++) {
  151     STATUS.innerText += CLASS_NAMES[n] + ' data count: ' + examplesCount[n] + '. ';
  152   }
  153   window.requestAnimationFrame(dataGatherLoop);
  154 }
  155 
```

Make your own "Teachable Machine" using Transfer Learning with MobileNet v3 in TensorFlow.js using saved graph model from TFHub.

Awaiting TF.js load

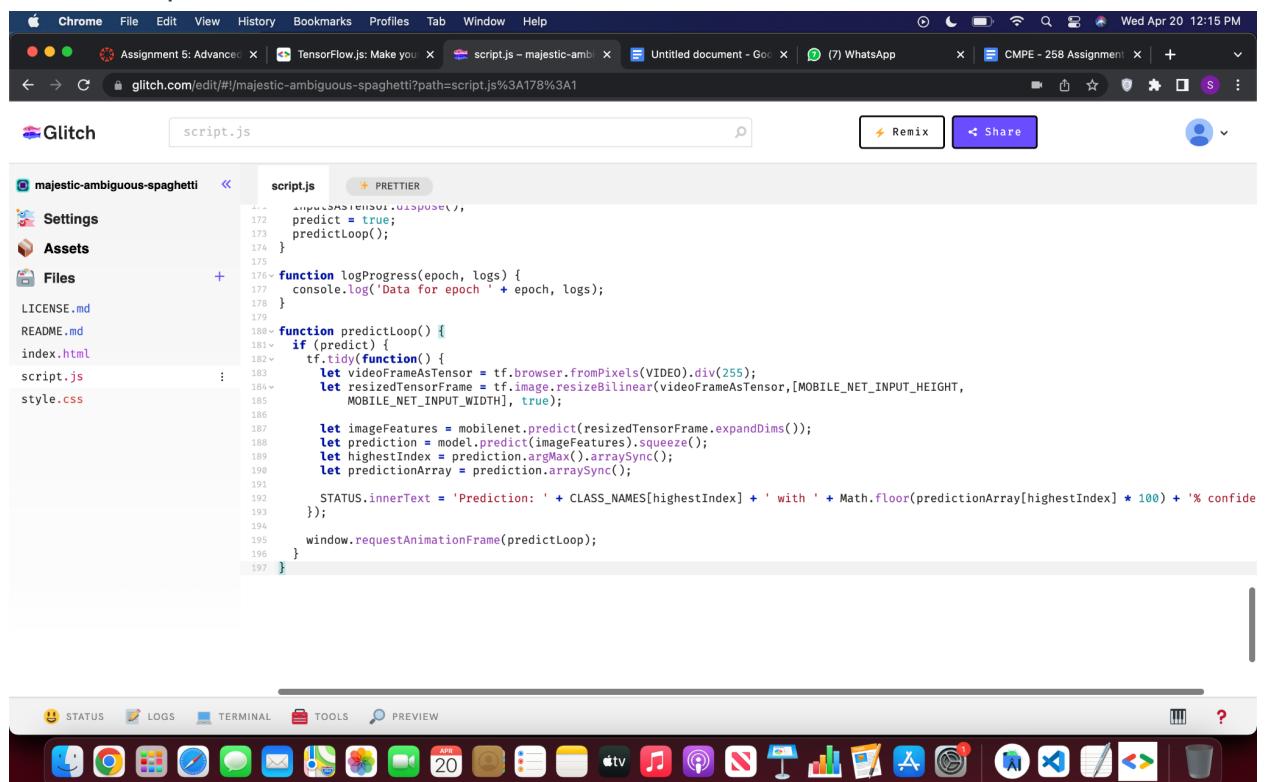
- Train and Predict



```

majestic-ambiguous-spaghetti <> script.js PRETTIER
Settings Assets Files LICENSE.md README.md index.html script.js style.css
  139 STATUS.innerHTML += CLASS_NAMES[n] + ' data count: ' + examplesCount[n] + '.';
  140 }
  141 window.requestAnimationFrame(dataGatherLoop);
  142 }
  143 }
  144 }
  145
  146 async function trainAndPredict() {
  147   predict = false;
  148   tf.util.shuffleCombo(trainingDataInputs, trainingDataOutputs);
  149   let outputsAsTensor = tf.tensorId(trainingDataOutputs, 'int32');
  150   let oneHotOutputs = tf.oneHot(outputsAsTensor, CLASS_NAMES.length);
  151   let inputsAsTensor = tf.stack(trainingDataInputs);
  152
  153   let results = await model.fit(inputsAsTensor, oneHotOutputs, {shuffle: true, batchSize: 5, epochs: 10,
  154     callbacks: {onEpochEnd: logProgress} });
  155
  156   outputsAsTensor.dispose();
  157   oneHotOutputs.dispose();
  158   inputsAsTensor.dispose();
  159   predict = true;
  160   predictLoop();
  161
  162   function logProgress(epoch, logs) {
  163     console.log('Data for epoch ' + epoch, logs);
  164   }
  165 }
```

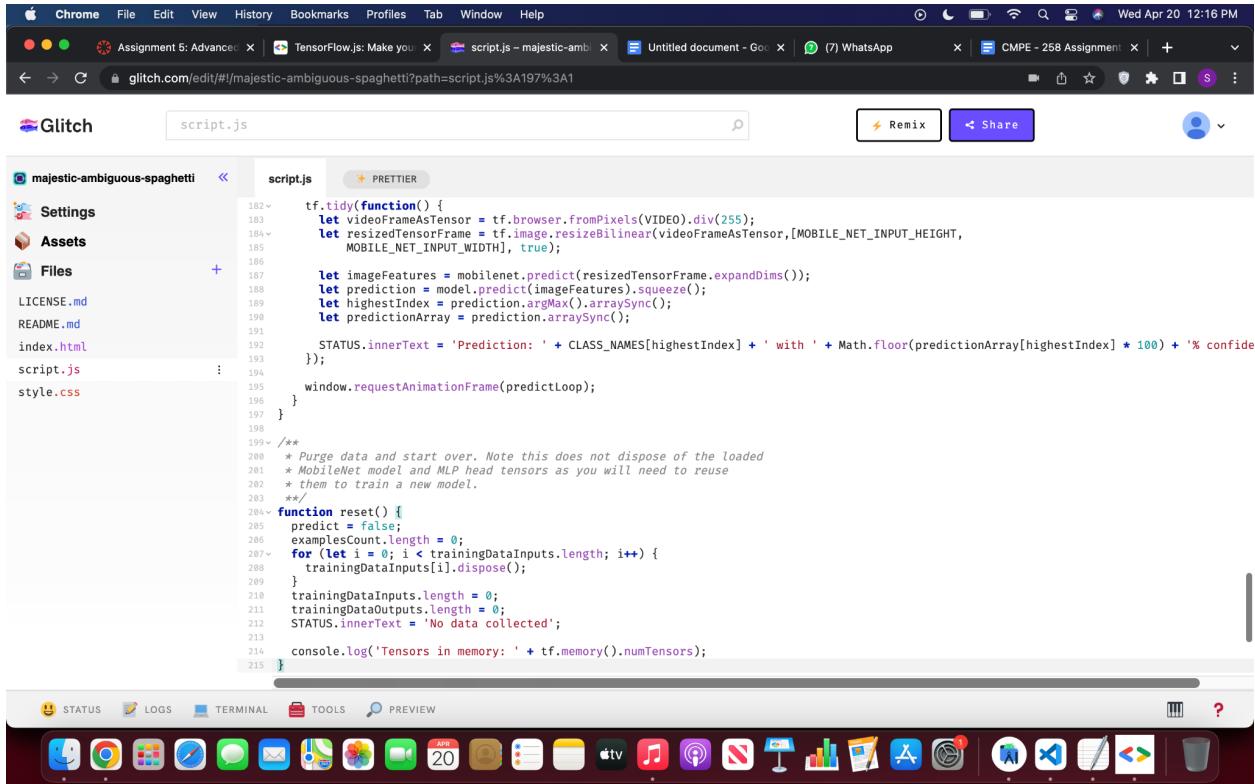
- Predict Loop



```

majestic-ambiguous-spaghetti <> script.js PRETTIER
Settings Assets Files LICENSE.md README.md index.html script.js style.css
  171   outputsAsTensor.dispose();
  172   predict = true;
  173   predictLoop();
  174 }
  175
  176 function logProgress(epoch, logs) {
  177   console.log('Data for epoch ' + epoch, logs);
  178 }
  179
  180 function predictLoop() {
  181   if (predict) {
  182     tf.tidy(function() {
  183       let videoFrameAsTensor = tf.browser.fromPixels(VIDEO).div(255);
  184       let resizedTensorFrame = tf.image.resizeBilinear(videoFrameAsTensor,[MOBILE_NET_INPUT_HEIGHT,
  185         MOBILE_NET_INPUT_WIDTH], true);
  186
  187       let imageFeatures = mobilenet.predict(resizedTensorFrame.expandDims());
  188       let prediction = model.predict(imageFeatures).squeeze();
  189       let highestIndex = prediction.argMax().arraySync();
  190       let predictionArray = prediction.arraySync();
  191
  192       STATUS.innerHTML = 'Prediction: ' + CLASS_NAMES[highestIndex] + ' with ' + Math.floor(predictionArray[highestIndex] * 100) + '% confidence';
  193     });
  194
  195     window.requestAnimationFrame(predictLoop);
  196   }
  197 }
```

- Implement the reset button



```

● Implement the reset button

script.js
PRETTIER

tf.tidy(function() {
  let videoFrameAsTensor = tf.browser.fromPixels(VIDEO).div(255);
  let resizedTensorFrame = tf.image.resizeBilinear(videoFrameAsTensor,[MOBILE_NET_INPUT_HEIGHT,
  MOBILE_NET_INPUT_WIDTH], true);

  let imageFeatures = mobilenet.predict(resizedTensorFrame.expandDims());
  let prediction = model.predict(imageFeatures).squeeze();
  let highestIndex = prediction.argmax().arraySync();
  let predictionArray = prediction.arraySync();

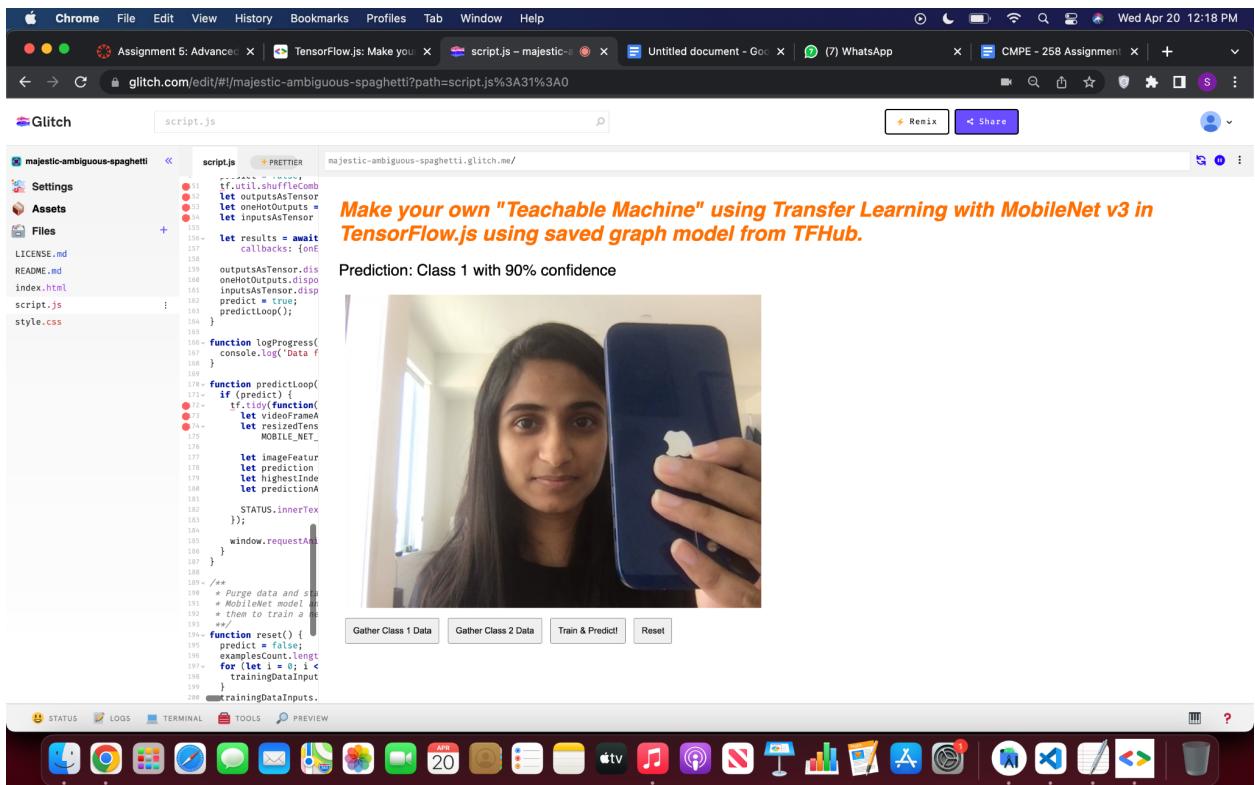
  STATUS.innerText = 'Prediction: ' + CLASS_NAMES[highestIndex] + ' with ' + Math.floor(predictionArray[highestIndex] * 100) + '% confidence';
}

window.requestAnimationFrame(predictLoop);
}

/*
 * Purge data and start over. Note this does not dispose of the loaded
 * MobileNet model and MLP head tensors as you will need to reuse
 * them to train a new model.
 */
function reset() {
predict = false;
examplesCount.length = 0;
for (let i = 0; i < trainingDataInputs.length; i++) {
  trainingDataInputs[i].dispose();
}
trainingDataInputs.length = 0;
trainingDataOutputs.length = 0;
STATUS.innerText = 'No data collected';
console.log('Tensors in memory: ' + tf.memory().numTensors);
}

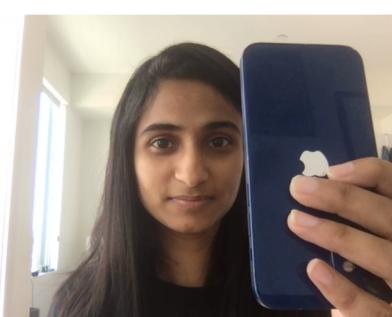

```

- Prediction Results



Make your own "Teachable Machine" using Transfer Learning with MobileNet v3 in TensorFlow.js using saved graph model from TFHub.

Prediction: Class 1 with 90% confidence



```

● Prediction Results

script.js PRETTIER
majestic-ambiguous-spaghetti/majestic-ambiguous-spaghetti.glitch.me

tf.tidy(function() {
  let videoFrameAsTensor = tf.browser.fromPixels(VIDEO).div(255);
  let resizedTensorFrame = tf.image.resizeBilinear(videoFrameAsTensor,[MOBILE_NET_INPUT_HEIGHT,
  MOBILE_NET_INPUT_WIDTH], true);

  let imageFeatures = mobilenet.predict(resizedTensorFrame.expandDims());
  let prediction = model.predict(imageFeatures).squeeze();
  let highestIndex = prediction.argmax().arraySync();
  let predictionArray = prediction.arraySync();

  STATUS.innerText = 'Prediction: ' + CLASS_NAMES[highestIndex] + ' with ' + Math.floor(predictionArray[highestIndex] * 100) + '% confidence';
}

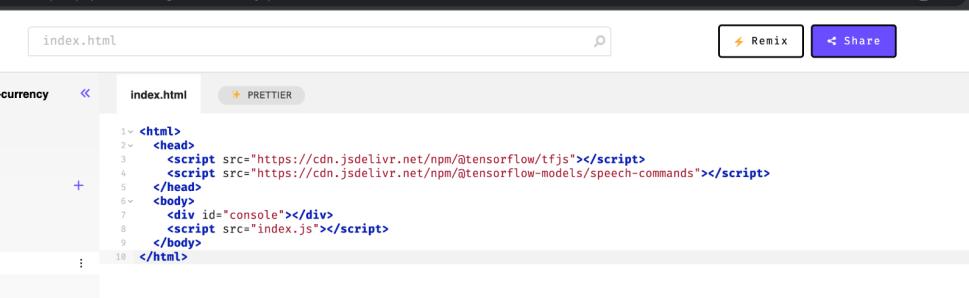
window.requestAnimationFrame(predictLoop);
}

/*
 * Purge data and start over. Note this does not dispose of the loaded
 * MobileNet model and MLP head tensors as you will need to reuse
 * them to train a new model.
 */
function reset() {
predict = false;
examplesCount.length = 0;
for (let i = 0; i < trainingDataInputs.length; i++) {
  trainingDataInputs[i].dispose();
}
trainingDataInputs.length = 0;
trainingDataOutputs.length = 0;
STATUS.innerText = 'No data collected';
console.log('Tensors in memory: ' + tf.memory().numTensors);
}


```

D) Webapp and audio training on device : TensorFlow.js - Audio recognition using transfer learning

- Load tensorflow.js and audio model



The screenshot shows a web browser window with several tabs open, including "Assignment 5: Advanced", "TensorFlow.js - Audio re...", "index.html - beneficial...", "Untitled document - Goo...", "(7) WhatsApp", and "CMPE - 258 Assignment". The main content area is a Glitch editor for a project named "beneficial-vigorous-currency". The left sidebar lists files: LICENSE.md, README.md, index.html, script.js, and style.css. The right sidebar shows the file structure and includes "Settings", "Assets", and "Files" sections. The central code editor displays the following HTML code:

```
<html>
  <head>
    <script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs"></script>
    <script src="https://cdn.jsdelivr.net/npm/@tensorflow-models/speech-commands"></script>
  </head>
  <body>
    <div id="console"></div>
    <script src="index.js"></script>
  </body>
</html>
```

- Create an index.js file and add the content to predict in real time



The image shows a Mac desktop environment. At the top, a browser window is open on the Glitch website, displaying a code editor for a project named "beneficial-vigorous-currency". The file "index.js" is selected, showing the following code:

```

1 let recognizer;
2
3 function predictWord() {
4   // Array of words that the recognizer is trained to recognize.
5   const words = recognizer.wordLabels();
6   recognizer.listen((scores) => {
7     // Turn scores into a list of (score,word) pairs.
8     scores = Array.from(scores).map((s, i) => ({score: s, word: words[i]}));
9     // Find the most probable word.
10    scores.sort((s1, s2) => s2.score - s1.score);
11    document.querySelector('#console').textContent = scores[0].word;
12  }, {probabilityThreshold: 0.75});
13
14
15 async function app() {
16   recognizer = speechCommands.create('BROWSER_FFT');
17   await recognizer.ensureModelLoaded();
18   predictWord();
19 }
20
21 app();

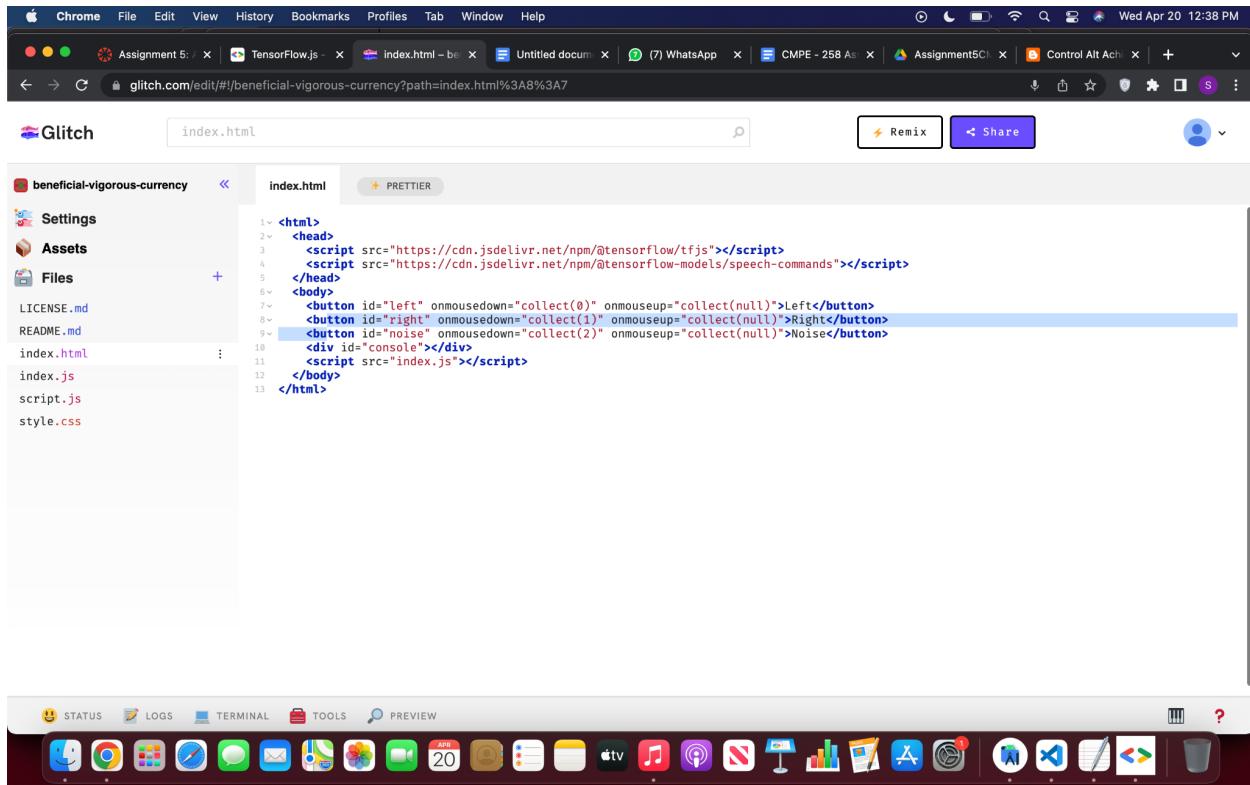
```

The browser window has tabs for "Assignment 5: Advanced", "TensorFlow.js - Audio re...", "index.js - beneficial-vigorous-currency", "Untitled document - Google Sheets", "(7) WhatsApp", and "CMPE - 258 Assignment". Below the browser is a file system view showing files like LICENSE.md, README.md, index.html, index.js, script.js, and style.css. On the right, there are "Remix" and "Share" buttons. The status bar at the bottom of the screen shows "Wed Apr 20 12:24 PM".

- Audio prediction results
- Saying the words are giving real time predictions

Drive link to the prediction results

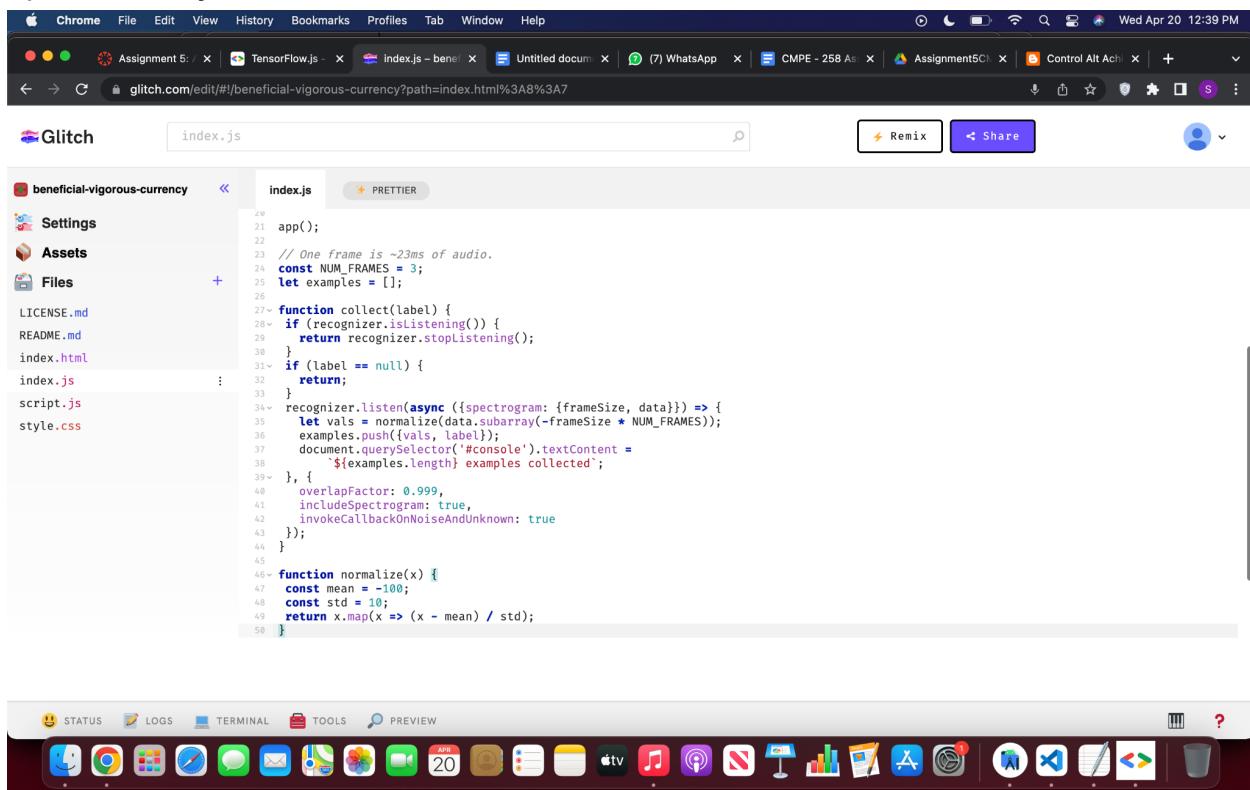
- Collect Data
- Update index.html



Glitch index.html PRETTIER

```
<html>
  <head>
    <script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs"></script>
    <script src="https://cdn.jsdelivr.net/npm/@tensorflow-models/speech-commands"></script>
  </head>
  <body>
    <button id="left" onmousedown="collect(0)" onmouseup="collect(null)">Left</button>
    <button id="right" onmousedown="collect(1)" onmouseup="collect(null)">Right</button>
    <button id="noise" onmousedown="collect(2)" onmouseup="collect(null)">Noise</button>
    <div id="console"></div>
    <script src="index.js"></script>
  </body>
</html>
```

Update index.js



Glitch index.js PRETTIER

```
app();
// One frame is ~23ms of audio.
const NUM_FRAMES = 3;
let examples = [];

function collect(label) {
  if (recognizer.isListening()) {
    return recognizer.stopListening();
  }
  if (label == null) {
    return;
  }
  recognizer.listen(async ({spectrogram: {frameSize, data}}) => {
    let vals = normalize(data.subarray(-frameSize * NUM_FRAMES));
    examples.push({vals, label});
    document.querySelector('#console').textContent =
      `${examples.length} examples collected`;
  }, {
    overlapFactor: 0.999,
    includeSpectrogram: true,
    invokeCallbackOnNoiseAndUnknown: true
  });
}

function normalize(x) {
  const mean = -100;
  const std = 10;
  return x.map(x => (x - mean) / std);
}
```

Remove predictWord() call

The screenshot shows a Glitch project interface. The top navigation bar includes tabs for Chrome, Edit, View, History, Bookmarks, Profiles, Tab, Window, Help, and a date/time indicator. Below the bar, a URL bar shows the address: glitch.com/edit/#/beneficial-vigorous-currency?path=index.js%3A50%3A1. The main workspace displays the `index.js` file with the following code:

```
function predictWord() {
  // Array of words that the recognizer is trained to recognize.
  const words = recognizer.wordLabels();
  recognizer.listen(({scores}) => {
    // Turn scores into a list of (score,word) pairs.
    const score = Array.from(scores).map((s, i) => ({score: s, word: words[i]}));
    // Find the most probable word.
    scores.sort((s1, s2) => s2.score - s1.score);
    document.querySelector('#console').textContent = scores[0].word;
  }, {probabilityThreshold: 0.75});
}

async function app() {
  recognizer = speechCommands.create('BROWSER_FFT');
  await recognizer.ensureModelLoaded();
  // predictWord();
}

app();

// One frame is ~23ms of audio.
const NUM_FRAMES = 3;
let examples = [];

function collect(label) {
  if (recognizer.isListening()) {
    return recognizer.stopListening();
  }
  if (label == null) {
    return;
  }
  recognizer.listen(async ({spectrogram: {frameSize, data}}) => {
    let vals = normalize(data.subarray(-frameSize * NUM_FRAMES));
    examples.push({vals, label});
  });
}
```

- Test Data Collection

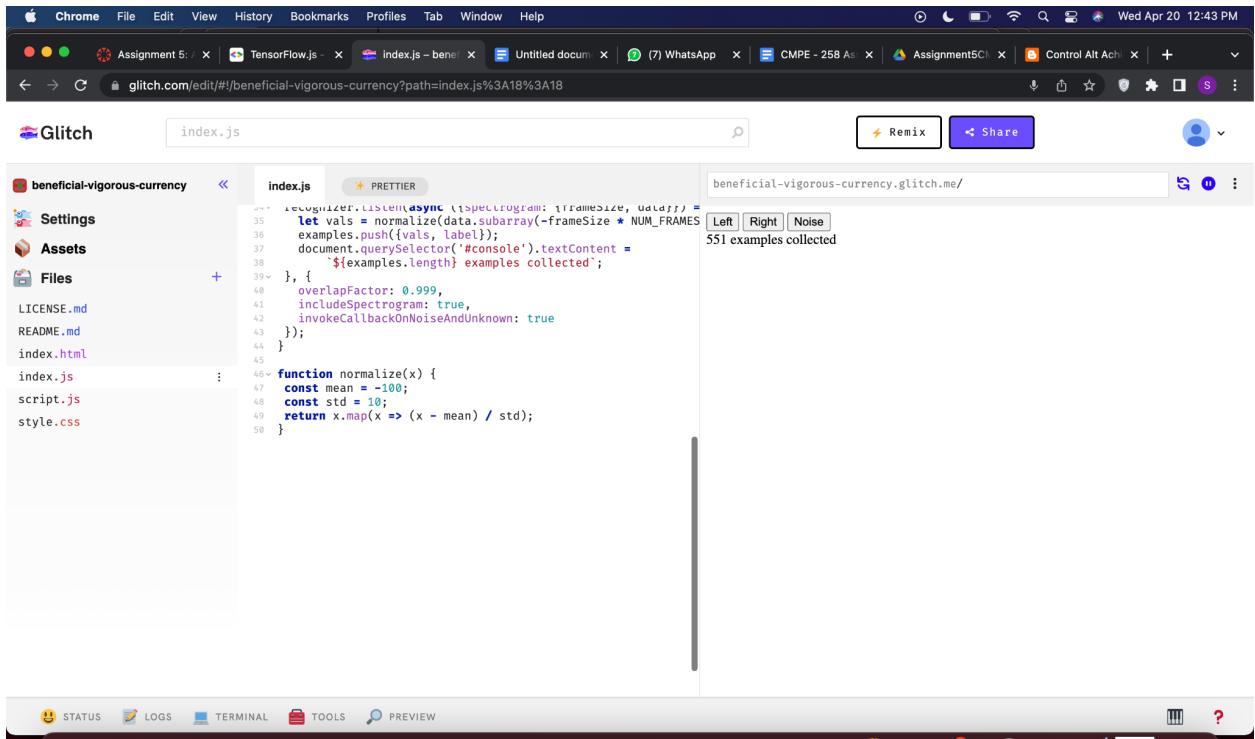
The screenshot shows a Glitch project titled "beneficial-vigorous-currency". The main editor window displays the file "index.js" with the following code:

```
recognizer.listen(async ({frameSize, update}) => {
  let vals = normalize(data.subarray(-frameSize * NUM_FRAMES));
  examples.push([vals, label]);
  document.querySelector('#console').textContent =
    `${examples.length} examples collected`;
}, {
  overlapFactor: 0.999,
  includeSpectrogram: true,
  invokeCallbackOnNoiseAndUnknown: true
});

function normalize(x) {
  const mean = -100;
  const std = 10;
  return x.map(x => (x - mean) / std);
}
```

The interface includes a sidebar with files like LICENSE.md, README.md, and index.html. A preview panel on the right shows a spectrogram with buttons for Left, Right, and Noise. The bottom navigation bar includes STATUS, LOGS, TERMINAL, TOOLS, and PREVIEW tabs.

- For Left, I have trained with clap sound



The screenshot shows a Mac desktop with a browser window open to glitch.com/edit/#/beneficial-vigorous-currency?path=index.js%3A18%3A18. The browser has multiple tabs open, including 'Assignment 5...', 'TensorFlow.js', 'index.js - bene', 'Untitled docm', '(7) WhatsApp', 'CMPE - 258 As...', 'Assignment5C...', 'Control Alt Act...', and 'Wed Apr 20 12:43 PM'. The main content area is a Glitch project titled 'beneficial-vigorous-currency'. On the left, there's a sidebar with 'Settings', 'Assets', and 'Files' sections, and files like 'LICENSE.md', 'README.md', 'index.html', 'index.js', 'script.js', and 'style.css'. The central area contains the 'index.js' code:

```

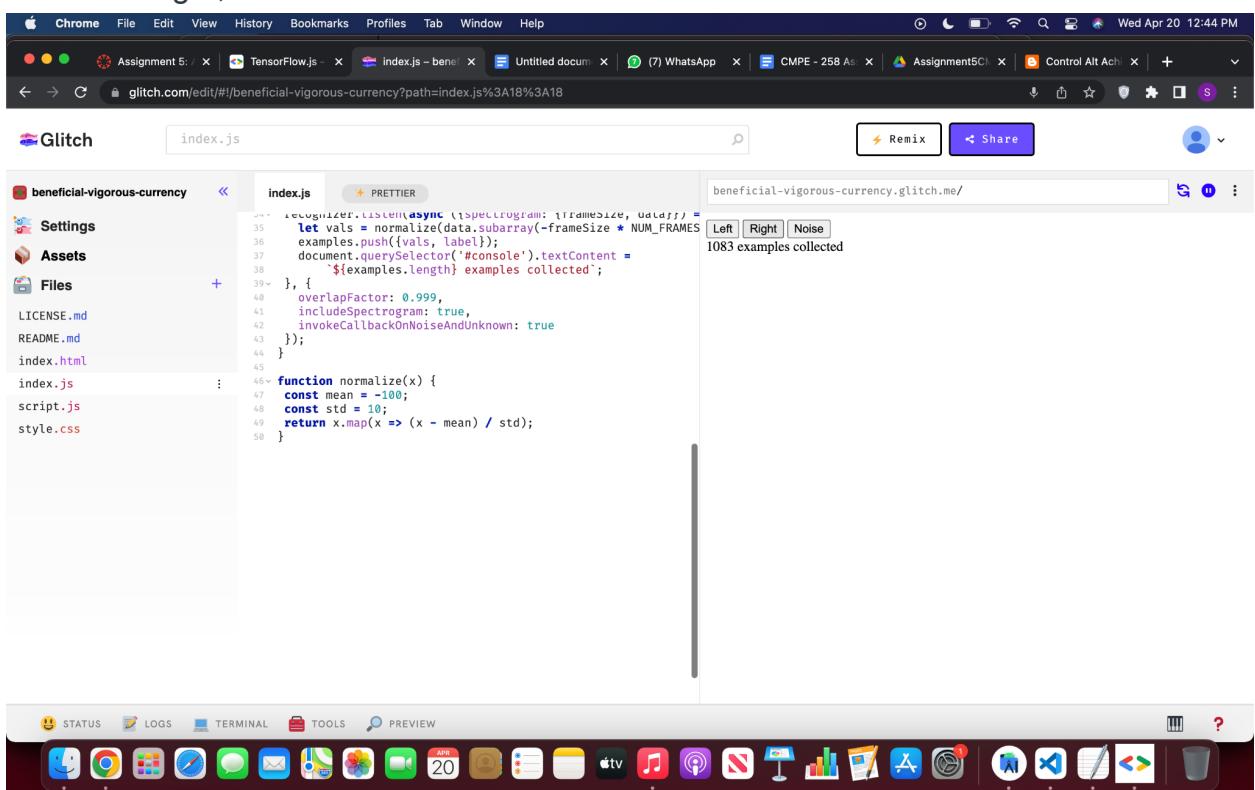
recognizer.listen(async (spectrogram, frameSize, overlapFactor) =>
  let vals = normalize(data.subarray(-frameSize * NUM_FRAMES));
  examples.push({vals, label});
  document.querySelector('#console').textContent =
    `${examples.length} examples collected`;
}, {
  overlapFactor: 0.999,
  includeSpectrogram: true,
  invokeCallbackOnNoiseAndUnknown: true
});

function normalize(x) {
  const mean = -100;
  const std = 10;
  return x.map(x => (x - mean) / std);
}

```

To the right, there's a preview tab titled 'beneficial-vigorous-currency.glitch.me' showing a spectrogram with three buttons: 'Left', 'Right', and 'Noise'. It says '551 examples collected'. Below the preview is a toolbar with icons for STATUS, LOGS, TERMINAL, TOOLS, and PREVIEW. The Mac OS X dock at the bottom contains various application icons.

- For right, I have trained with whistle sound



This screenshot is nearly identical to the one above, showing the same Glitch project and code. The difference is in the preview tab, which now shows a spectrogram with the 'Right' button selected. It says '1083 examples collected'. The rest of the interface, including the sidebar, code editor, and dock, remains the same.

- For noise, I have trained with talking

The screenshot shows a Glitch project titled "beneficial-vigorous-currency". The code editor displays "index.js" with Prettier styling applied. The file contains JavaScript code for a speech recognition model. A sidebar shows project files like LICENSE.md, README.md, index.html, index.js, script.js, and style.css. On the right, there's a preview window showing the project URL "beneficial-vigorous-currency.glitch.me/" and a status bar indicating "1710 examples collected". Below the browser is a Mac OS X dock with various application icons.

```

    recognizer.listen(async (spectrogram, frameSize, numFrames) => {
  let vals = normalize(data.subarray(-frameSize * numFrames));
  examples.push({vals, label});
  document.querySelector("#console").textContent =
    `${examples.length} examples collected`;
}, {
  overlapFactor: 0.999,
  includeSpectrogram: true,
  invokeCallbackOnNoiseAndUnknown: true
});
}

function normalize(x) {
  const mean = -100;
  const std = 10;
  return x.map(x => (x - mean) / std);
}
  
```

- Train a model

Add train button right after noise button

Glitch

index.html

```
<html>
  <head>
    <script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs"></script>
    <script src="https://cdn.jsdelivr.net/npm/@tensorflow-models/speech-commander"></script>
  </head>
  <body>
    <button id="left" onmousedown="collect(0)" onmouseup="collect(null)">Left</button>
    <button id="right" onmousedown="collect(1)" onmouseup="collect(null)">Right</button>
    <button id="noise" onmousedown="collect(2)" onmouseup="collect(null)">Noise</button>
    <br/><br/>
    <button id="train" onclick="train()">Train</button>
    <div id="console"></div>
    <script src="index.js"></script>
  </body>
</html>
```

beneficial-vigorous-currency.glitch.me/

Left Right Noise

Train

STATUS LOGS TERMINAL TOOLS PREVIEW

Update index.js

Glitch

index.js

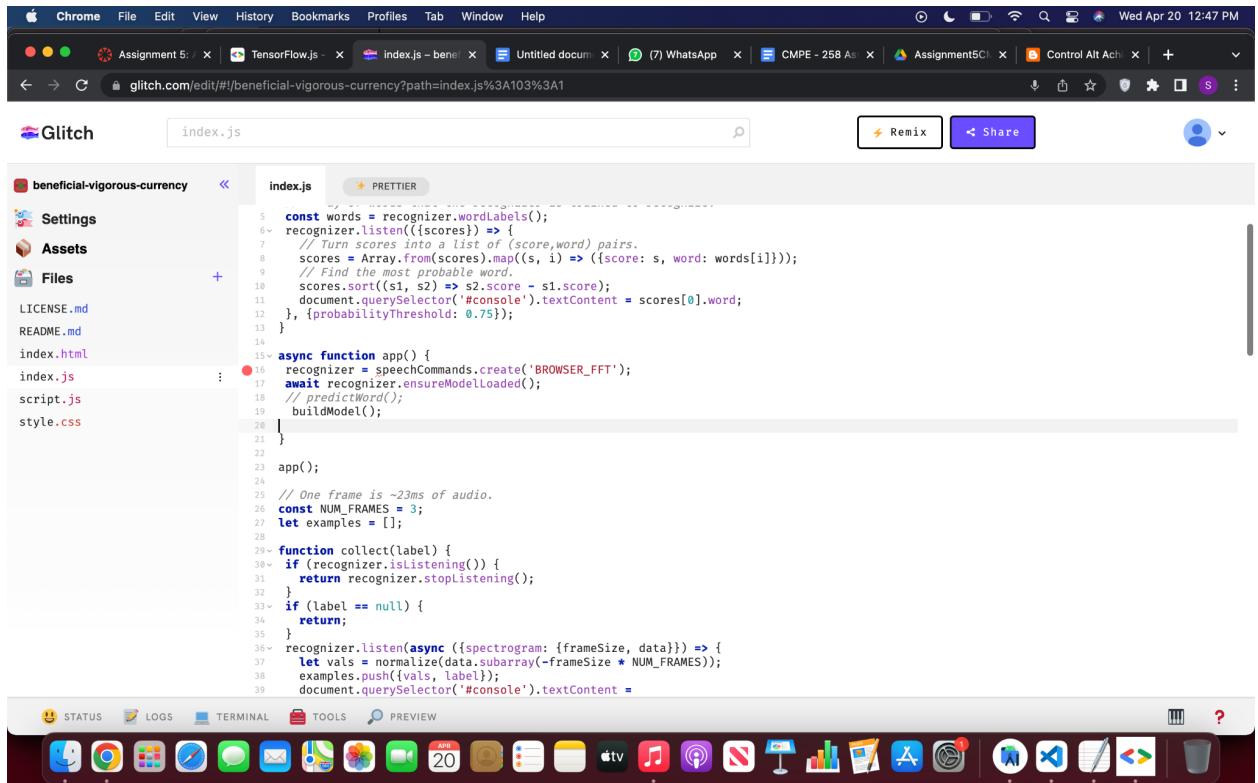
```
function buildModel() {
  model = tf.sequential();
  model.add(tf.layers.depthwiseConv2d({
    depthMultiplier: 8,
    kernelSize: [NUM_FRAMES, 3],
    activation: 'relu',
    inputShape: INPUT_SHAPE
  }));
  model.add(tf.layers.maxPooling2d({poolSize: [1, 2], strides: [2, 2]}));
  model.add(tf.layers.flatten());
  model.add(tf.layers.dense({units: 3, activation: 'softmax'}));
  const optimizer = tf.train.adam(0.01);
  model.compile({
    optimizer,
    loss: 'categoricalCrossentropy',
    metrics: ['accuracy']
  });
}

function toggleButtons(enable) {
  document.querySelectorAll('button').forEach(b => b.disabled = !enable);
}

function flatten(tensors) {
  const size = tensors[0].length;
  const result = new Float32Array(tensors.length * size);
  tensors.forEach((arr, i) => result.set(arr, i * size));
  return result;
}
```

STATUS LOGS TERMINAL TOOLS PREVIEW

Call buildModel() when app loads



A screenshot of a Mac desktop showing a browser window on glitch.com/edit/#/beneficial-vigorous-currency?path=index.js%3A103%3A1. The window title is "Assignment 5...". The page displays the contents of index.js:

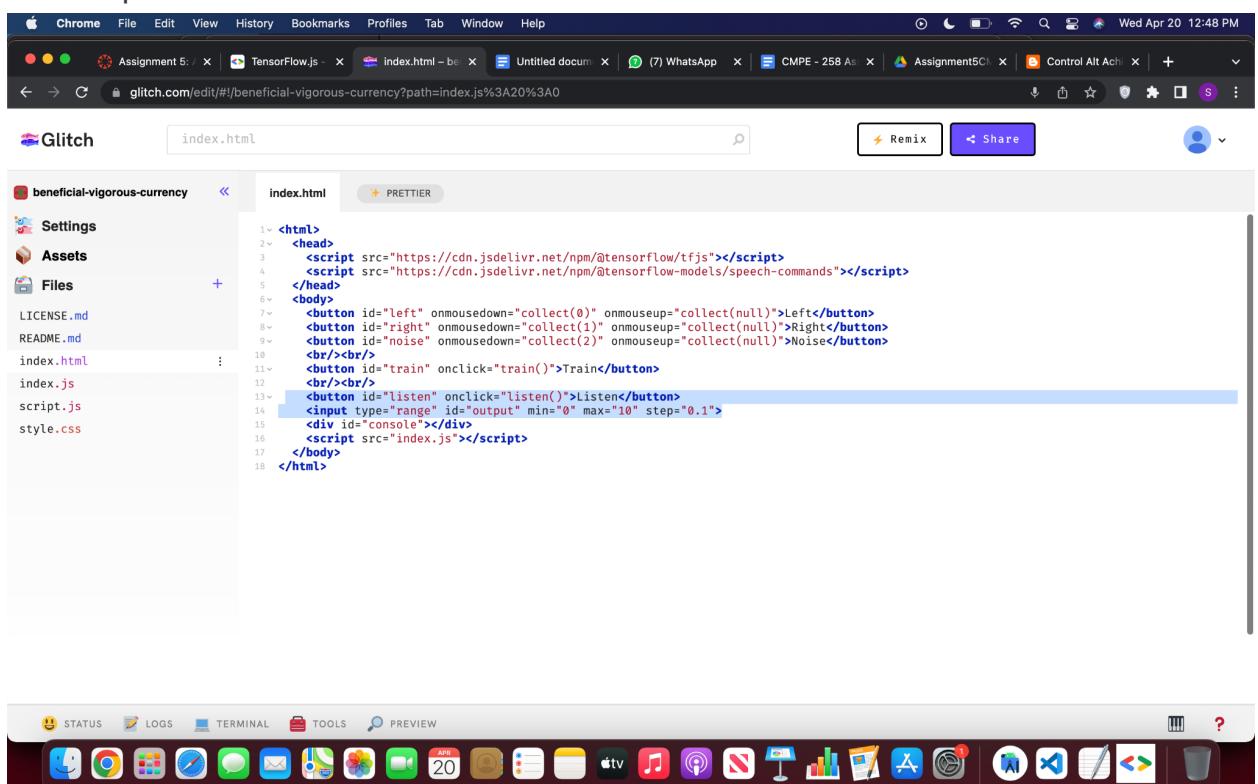
```
const words = recognizer.wordLabels();
recognizer.listen((scores) => {
  // Turn scores into a list of (score, word) pairs.
  scores = Array.from(scores).map((s, i) => ({score: s, word: words[i]}));
  // Find the most probable word.
  scores.sort((s1, s2) => s2.score - s1.score);
  document.querySelector('#console').textContent = scores[0].word;
}, {probabilityThreshold: 0.75});

async function app() {
  recognizer = speechCommands.create('BROWSER_FFT');
  await recognizer.ensureModelLoaded();
  // predictWord();
  buildModel();
}

// One frame is ~23ms of audio.
const NUM_FRAMES = 3;
let examples = [];

function collect(label) {
  if (recognizer.isListening()) {
    return recognizer.stopListening();
  }
  if (label == null) {
    return;
  }
  recognizer.listen(async ({spectrogram: {frameSize, data}}) => {
    let vals = normalize(data.subarray(-frameSize * NUM_FRAMES));
    examples.push({vals, label});
    document.querySelector('#console').textContent =
      `Collected ${examples.length} examples`;
  });
}
```

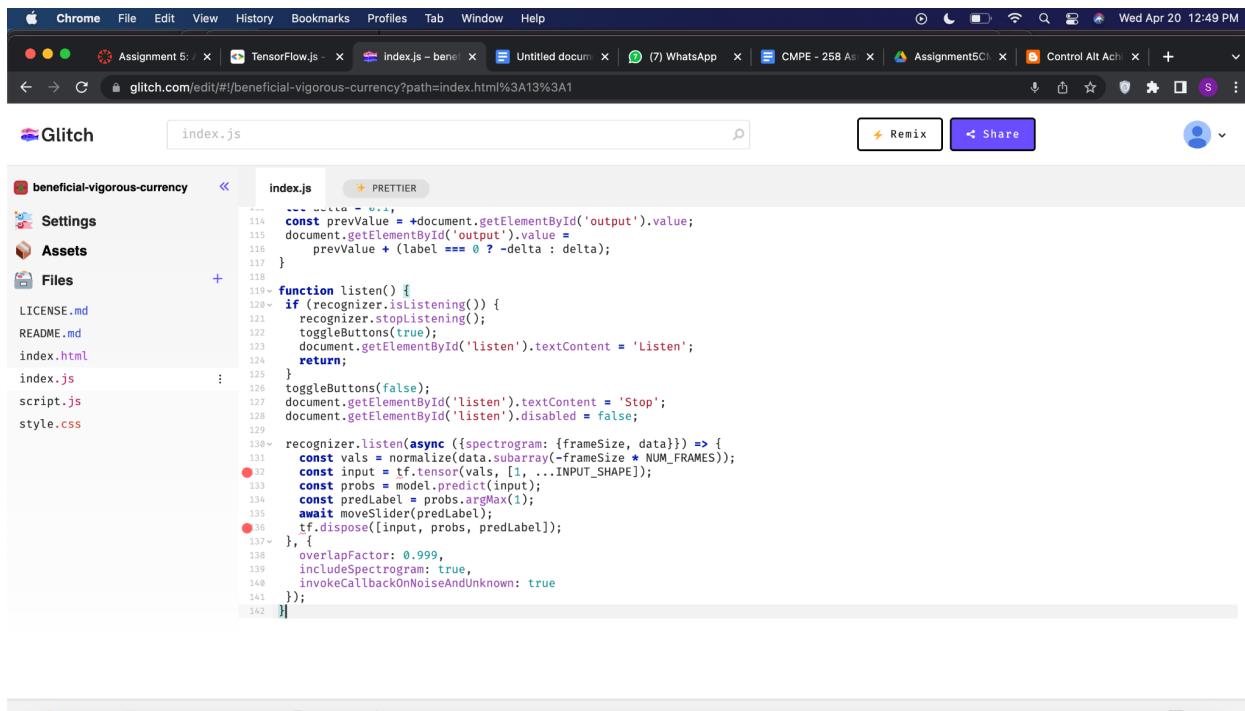
● Update slider in real time



A screenshot of a Mac desktop showing a browser window on glitch.com/edit/#/beneficial-vigorous-currency?path=index.js%3A20%3A0. The window title is "Assignment 5...". The page displays the contents of index.html:

```
<html>
  <head>
    <script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs"></script>
    <script src="https://cdn.jsdelivr.net/npm/@tensorflow-models/speech-commands"></script>
  </head>
  <body>
    <button id="left" onmousedown="collect(0)" onmouseup="collect(null)">Left</button>
    <button id="right" onmousedown="collect(1)" onmouseup="collect(null)">Right</button>
    <button id="noise" onmousedown="collect(2)" onmouseup="collect(null)">Noise</button>
    <br/><br/>
    <button id="train" onclick="train()">Train</button>
    <br/><br/>
    <button id="listen" onclick="listen()">Listen</button>
    <input type="range" id="output" min="0" max="10" step="0.1">
    <div id="console"></div>
    <script src="index.js"></script>
  </body>
</html>
```

Update index.js

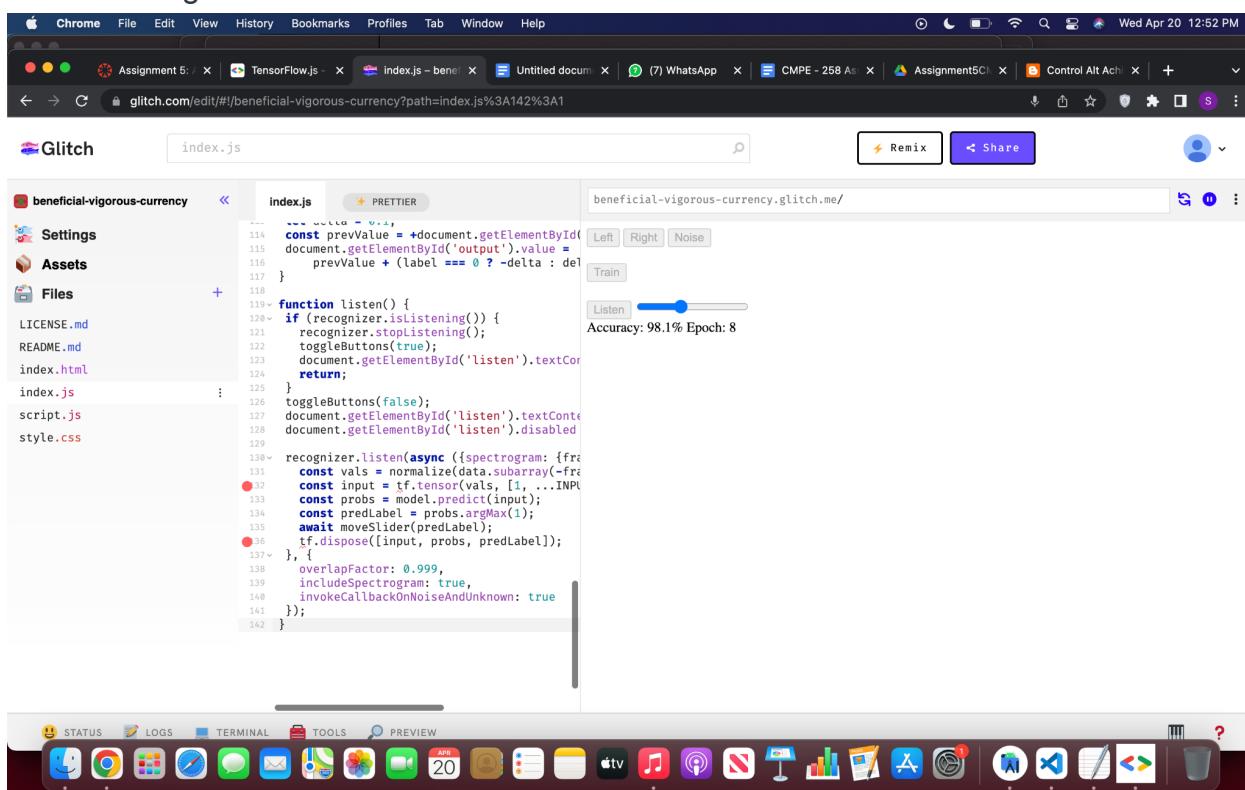


```
const recognizer = new webkitSpeechRecognition();
const prevValue = +document.getElementById('output').value;
document.getElementById('output').value =
    prevValue + (label === 0 ? -delta : delta);
}

function listen() {
    if (recognizer.isListening()) {
        recognizer.stopListening();
        toggleButtons(true);
        document.getElementById('listen').textContent = 'Listen';
        return;
    }
    toggleButtons(false);
    document.getElementById('listen').textContent = 'Stop';
    document.getElementById('listen').disabled = false;
}

recognizer.listen(async ({spectrogram: {frameSize, data}}) => {
    const vals = normalize(data.subarray(-frameSize * NUM_FRAMES));
    const input = tf.tensor(vals, [1, ...INPUT_SHAPE]);
    const probs = model.predict(input);
    const predLabel = probs.argMax(1);
    await moveSlider(predLabel);
    tf.dispose([input, probs, predLabel]);
}, {
    overlapFactor: 0.999,
    includeSpectrogram: true,
    invokeCallbackOnNoiseAndUnknown: true
});
}
```

● Testing the final result

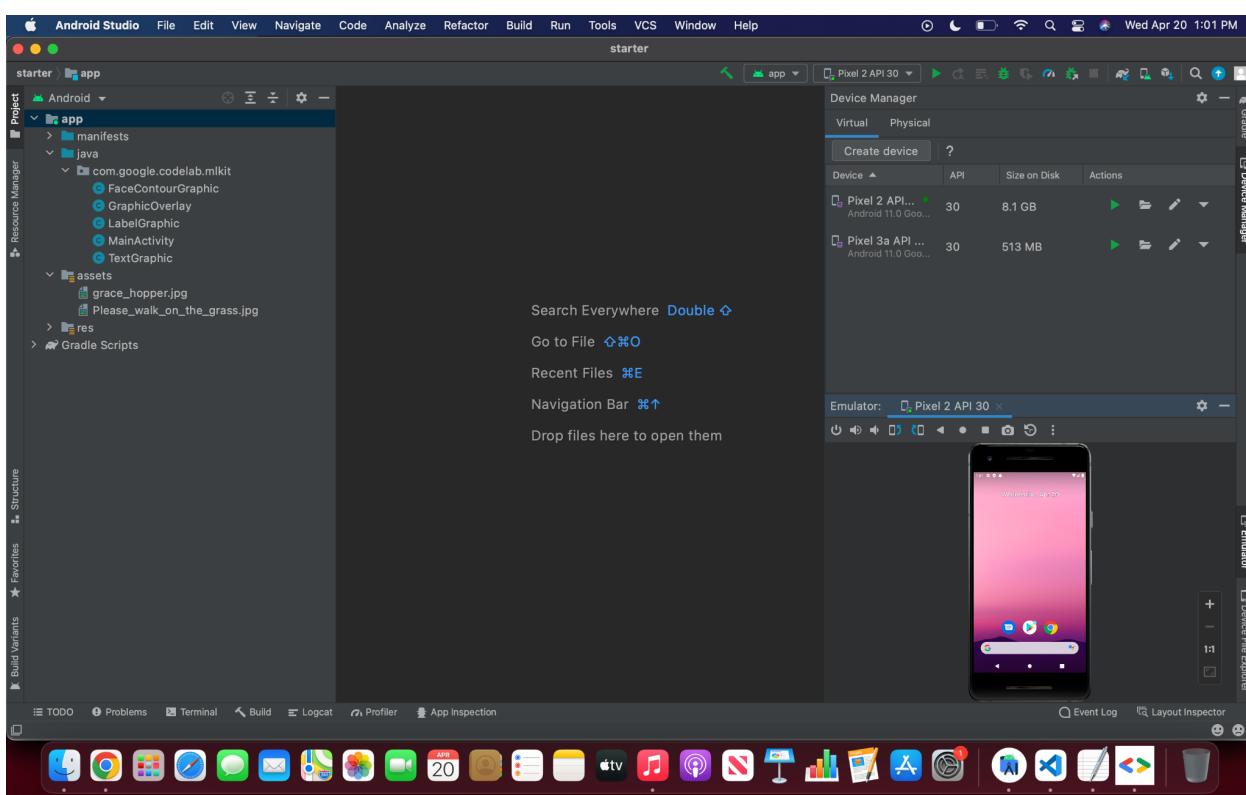
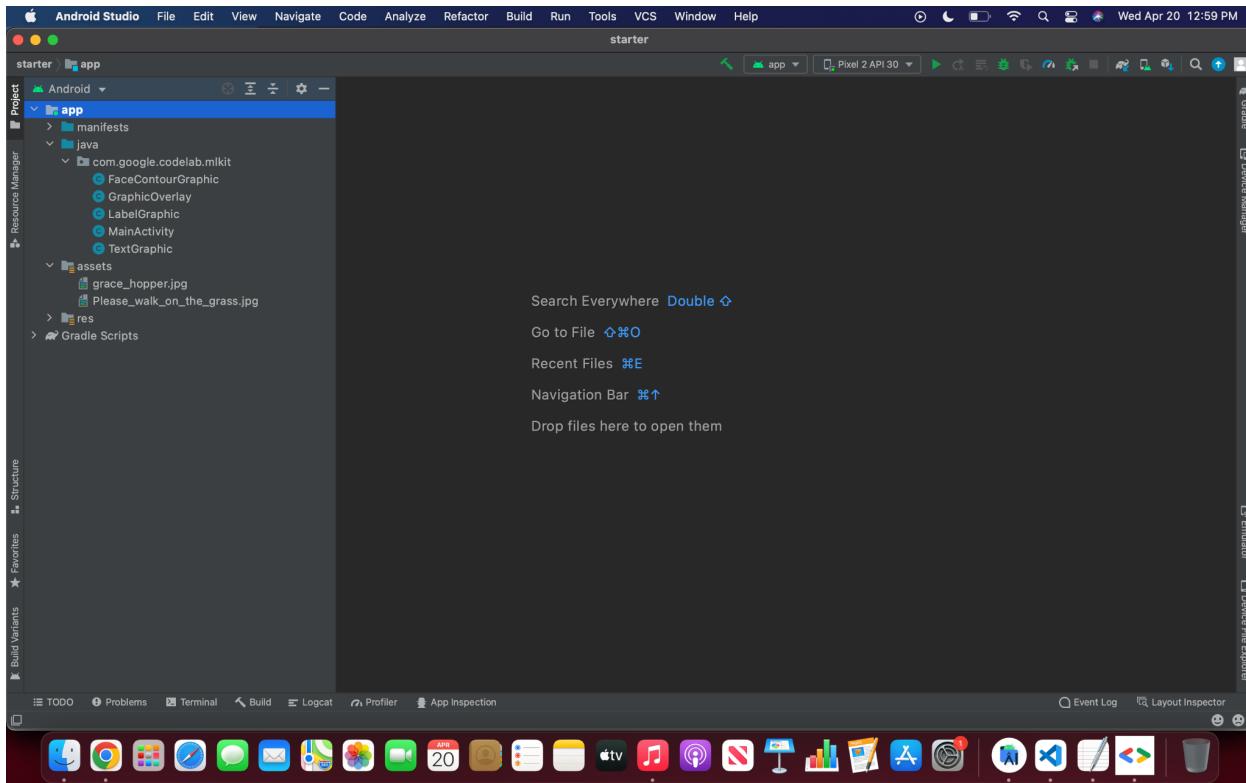


```
const recognizer = new webkitSpeechRecognition();
const prevValue = +document.getElementById('output').value;
document.getElementById('output').value =
    prevValue + (label === 0 ? -delta : delta);
}

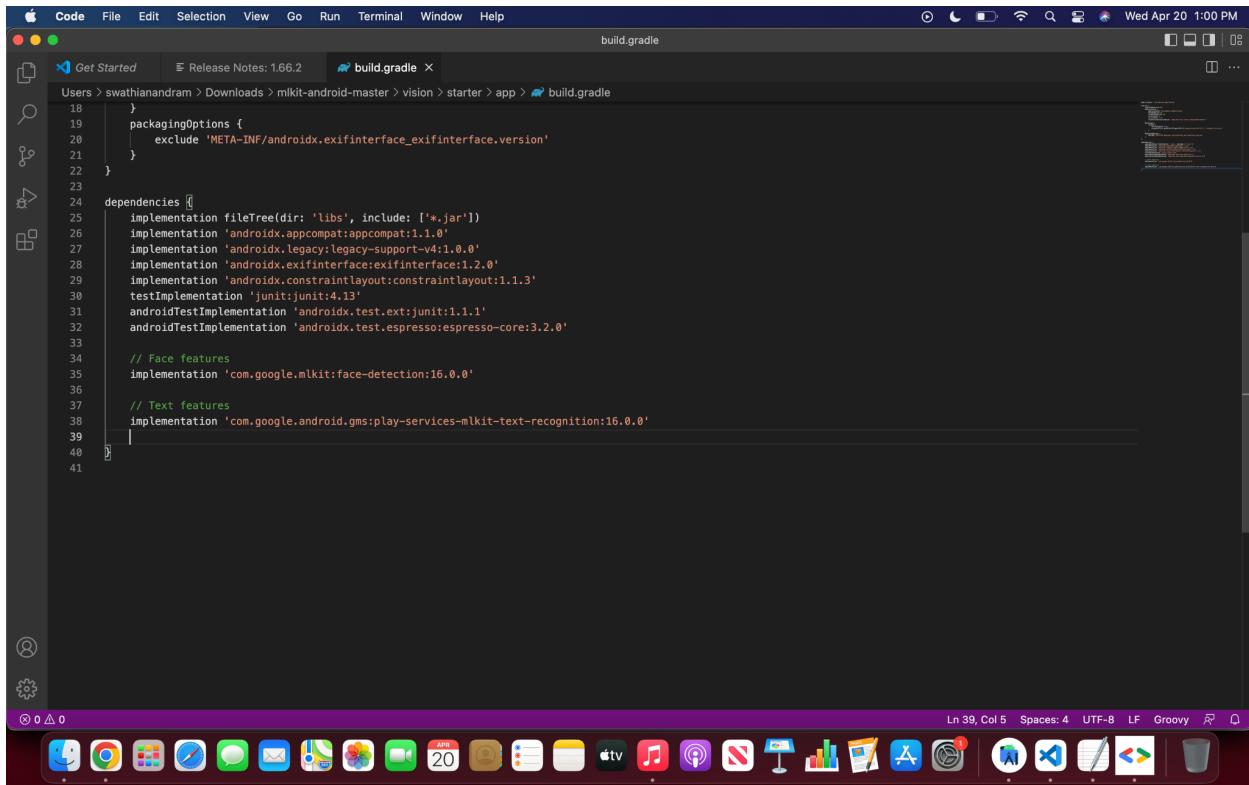
function listen() {
    if (recognizer.isListening()) {
        recognizer.stopListening();
        toggleButtons(true);
        document.getElementById('listen').textContent = 'Listen';
        return;
    }
    toggleButtons(false);
    document.getElementById('listen').textContent = 'Stop';
    document.getElementById('listen').disabled = false;
}

recognizer.listen(async ({spectrogram: {frameSize, data}}) => {
    const vals = normalize(data.subarray(-frameSize * NUM_FRAMES));
    const input = tf.tensor(vals, [1, ...INPUT_SHAPE]);
    const probs = model.predict(input);
    const predLabel = probs.argMax(1);
    await moveSlider(predLabel);
    tf.dispose([input, probs, predLabel]);
}, {
    overlapFactor: 0.999,
    includeSpectrogram: true,
    invokeCallbackOnNoiseAndUnknown: true
});
```

- E) Using out of box sdk to do ML ondevice : Use ML Kit to perform Recognize text and facial features with ML Kit: Android and Recognize, Identify Language and Translate text with ML Kit and CameraX: Android
 - Setting up the project on Android Studio

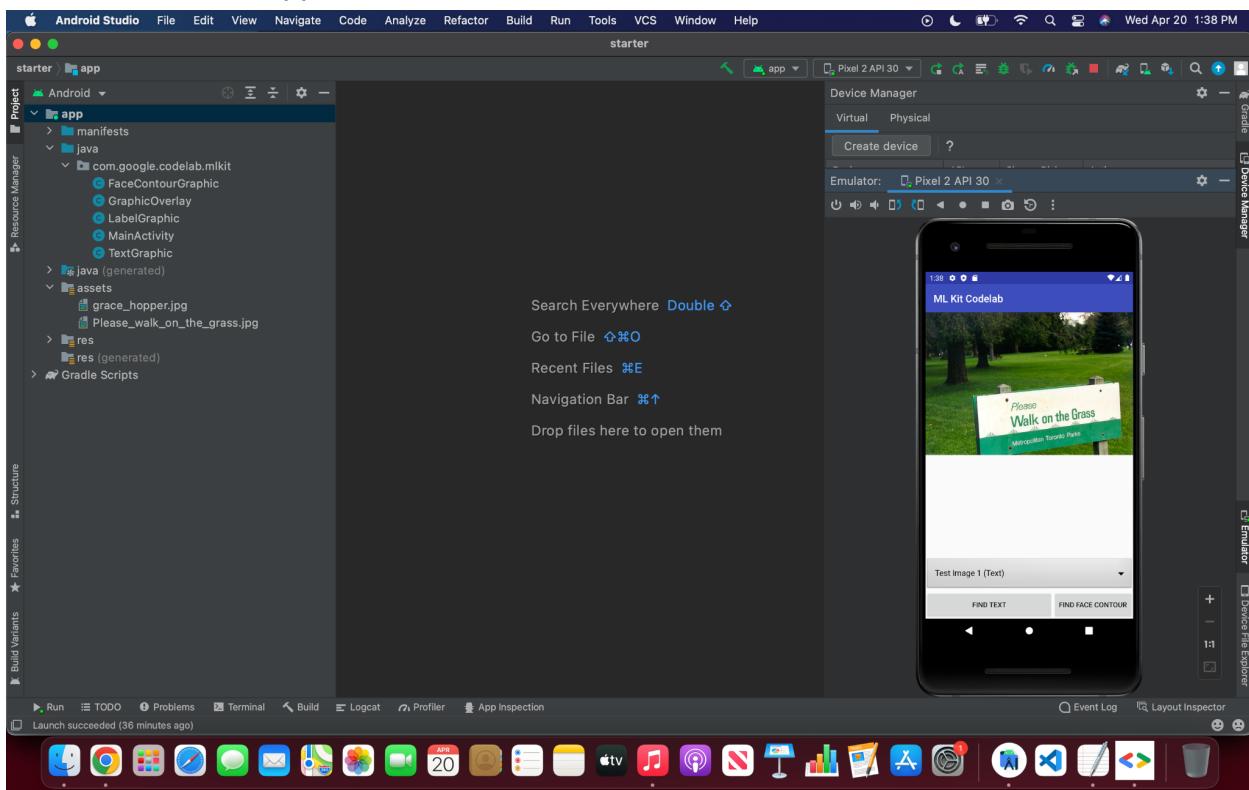


- Checking necessary dependencies

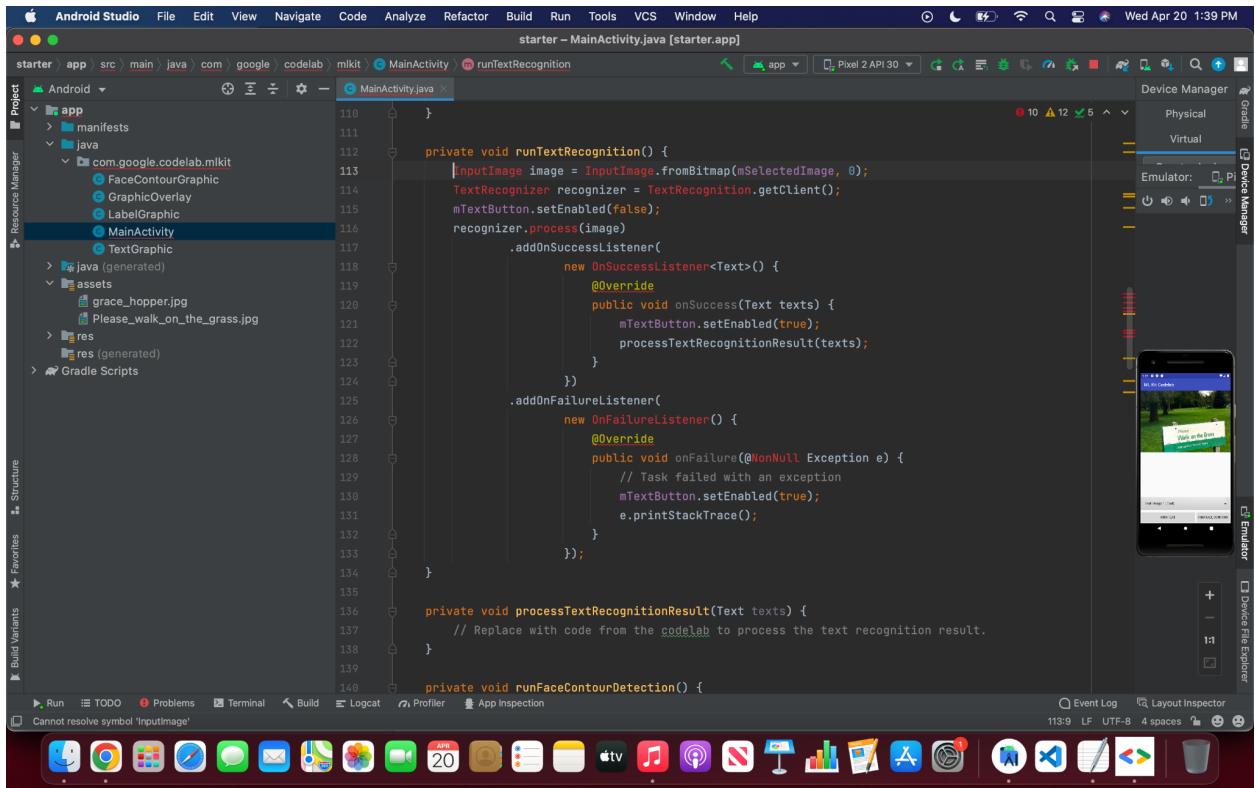


```
build.gradle
Users > swathianandram > Downloads > mlkit-android-master > vision > starter > app > build.gradle
18     }
19     packagingOptions {
20         exclude 'META-INF/androidx.exifinterface_exifinterface.version'
21     }
22 }
23
24 dependencies {
25     implementation fileTree(dir: 'libs', include: ['*.jar'])
26     implementation 'androidx.appcompat:appcompat:1.1.0'
27     implementation 'androidx.legacy:legacy-support-v4:1.0.0'
28     implementation 'androidx.exifinterface:exifinterface:1.2.0'
29     implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
30     testImplementation 'junit:junit:4.13'
31     androidTestImplementation 'androidx.test.ext:junit:1.1.1'
32     androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0'
33
34     // Face features
35     implementation 'com.google.mlkit:face-detection:16.0.0'
36
37     // Text features
38     implementation 'com.google.android.gms:play-services-mlkit-text-recognition:16.0.0'
39
40
41 }
```

- Run starter app



- Update runTextRecognition function



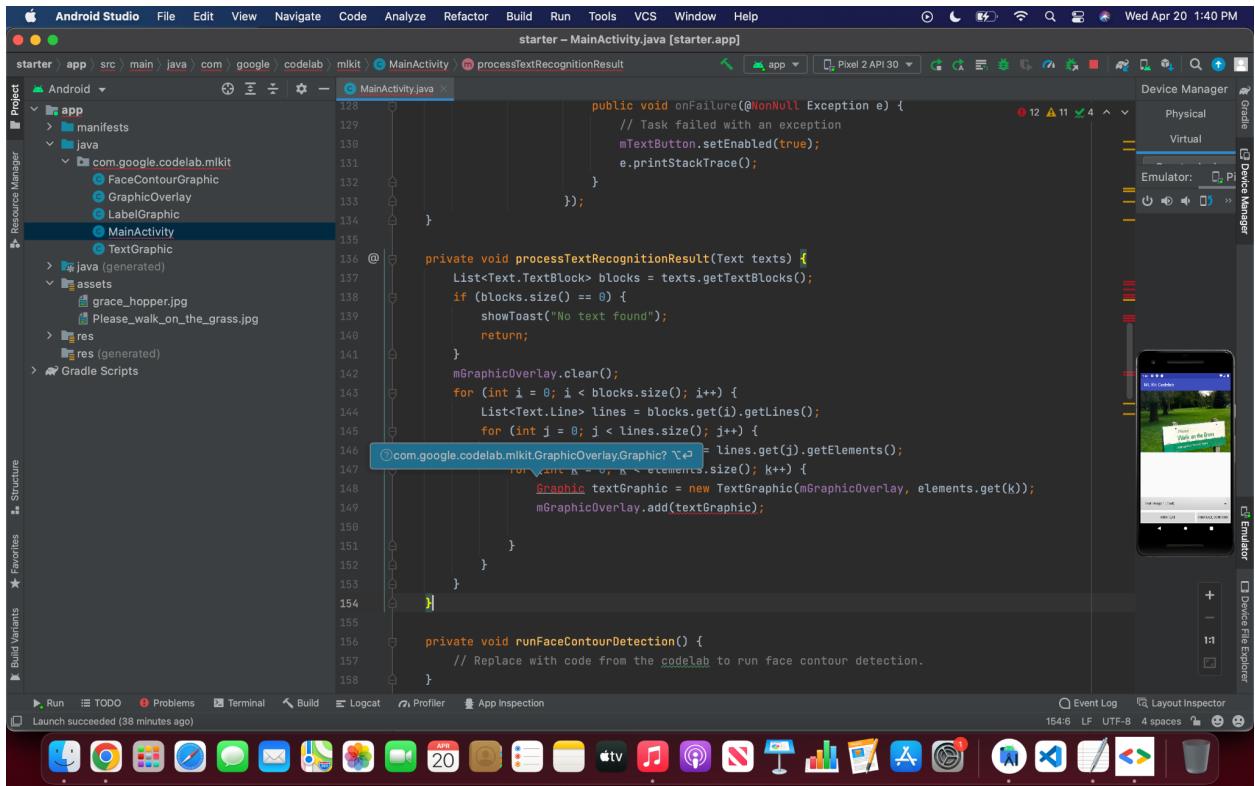
The screenshot shows the Android Studio interface with the project 'starter' open. The main window displays the Java file `MainActivity.java`. The code implements the `runTextRecognition` method, which uses the `TextRecognition` API to process an image and add success/failure listeners. The code also includes a placeholder for `processTextRecognitionResult`. On the right side, there's a preview of the app running on an emulator showing a scene from 'Beauty and the Beast' with text overlays.

```

110     }
111
112     private void runTextRecognition() {
113         InputImage image = InputImage.fromBitmap(mSelectedImage, 0);
114         TextRecognizer recognizer = TextRecognition.getClient();
115         mTextButton.setEnabled(false);
116         recognizer.process(image)
117             .addOnSuccessListener(
118                 new OnSuccessListener<Text>() {
119                     @Override
120                     public void onSuccess(Text texts) {
121                         mTextButton.setEnabled(true);
122                         processTextRecognitionResult(texts);
123                     }
124                 })
125             .addOnFailureListener(
126                 new OnFailureListener() {
127                     @Override
128                     public void onFailure(@NonNull Exception e) {
129                         // Task failed with an exception
130                         mTextButton.setEnabled(true);
131                         e.printStackTrace();
132                     }
133                 });
134     }
135
136     private void processTextRecognitionResult(Text texts) {
137         // Replace with code from the code lab to process the text recognition result.
138     }
139
140     private void runFaceContourDetection() {
141
142     }

```

- Update processTextRecognitionResult() function



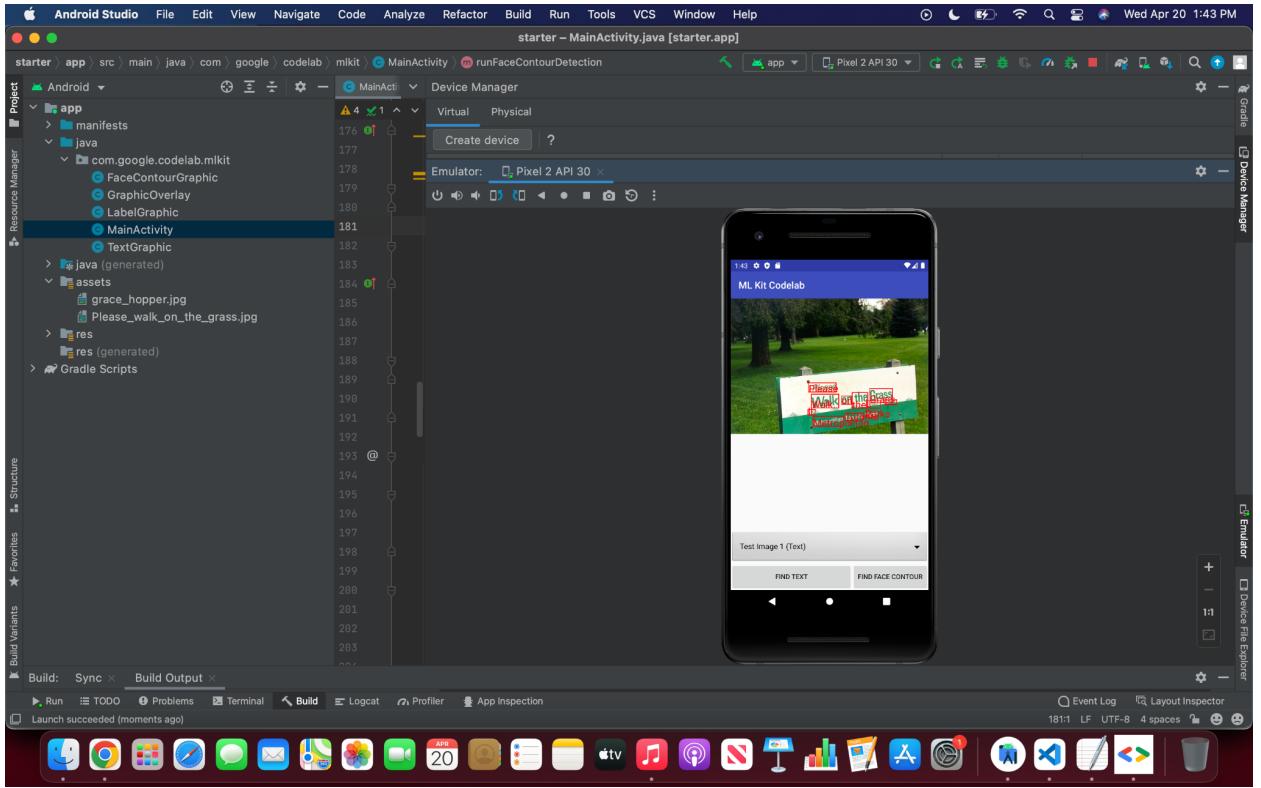
The screenshot shows the Android Studio interface with the project 'starter' open. The main window displays the Java file `MainActivity.java`. The code implements the `processTextRecognitionResult` method, which retrieves text blocks from the `Text` object and iterates through them to create `TextGraphic` objects and add them to a `GraphicOverlay`. A placeholder for `runFaceContourDetection` is also present. The right side shows the app running on an emulator with text overlays.

```

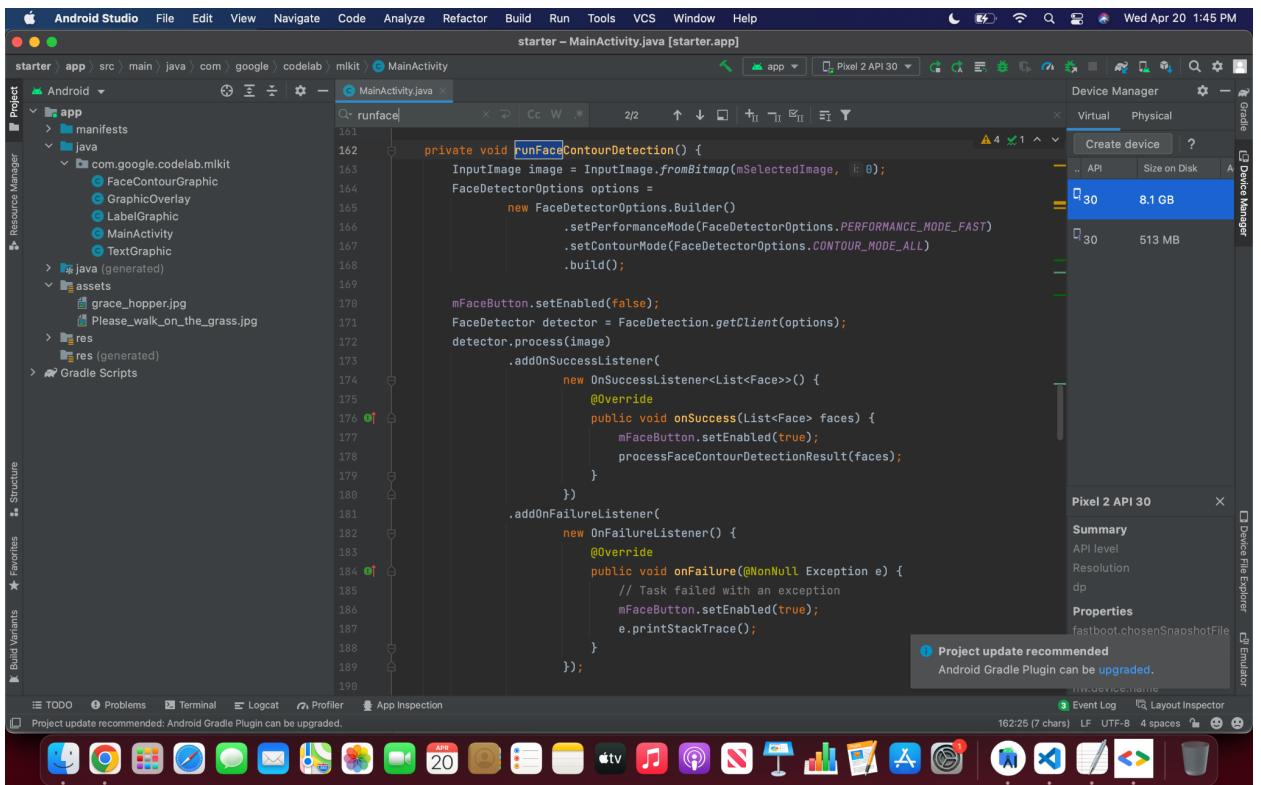
128     public void onFailure(@NonNull Exception e) {
129         // Task failed with an exception
130         mTextButton.setEnabled(true);
131         e.printStackTrace();
132     }
133 }
134
135 private void processTextRecognitionResult(Text texts) {
136     List<Text.TextBlock> blocks = texts.getTextBlocks();
137     if (blocks.size() == 0) {
138         showToast("No text found");
139         return;
140     }
141     mGraphicOverlay.clear();
142     for (int i = 0; i < blocks.size(); i++) {
143         List<Text.Line> lines = blocks.get(i).getLines();
144         for (int j = 0; j < lines.size(); j++) {
145             com.google.codelab.mlkit.Graphic? lines.get(j).getElements();
146             for (int k = 0; k < elements.size(); k++) {
147                 Graphic textGraphic = new TextGraphic(mGraphicOverlay, elements.get(k));
148                 mGraphicOverlay.add(textGraphic);
149             }
150         }
151     }
152 }
153
154 private void runFaceContourDetection() {
155     // Replace with code from the code lab to run face contour detection.
156 }
157
158

```

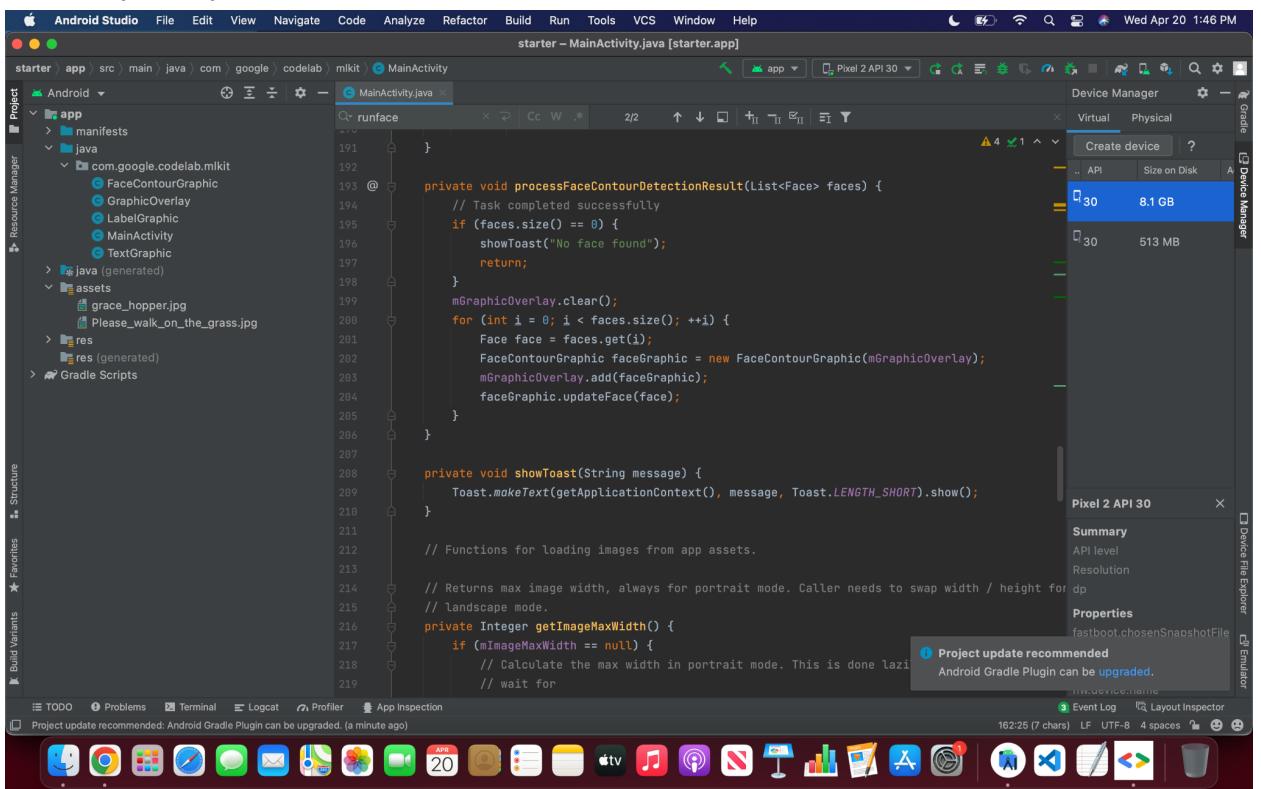
- Run the app on emulator and click find text



- Add on device face contour detection



- Update process face contour detection result function



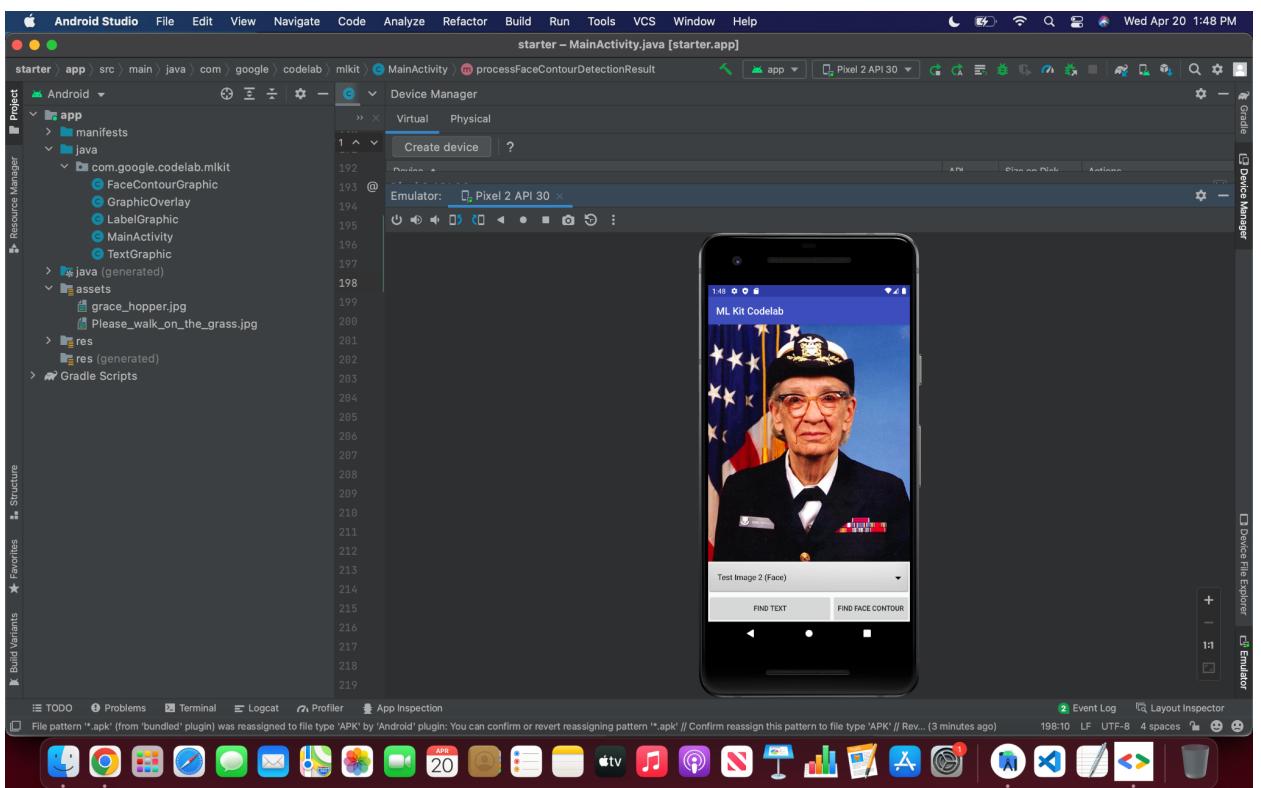
```

191     }
192
193     @Override
194     private void processFaceContourDetectionResult(List<Face> faces) {
195         // Task completed successfully
196         if (faces.size() == 0) {
197             showToast("No face found");
198             return;
199         }
200         mGraphicOverlay.clear();
201         for (int i = 0; i < faces.size(); ++i) {
202             Face face = faces.get(i);
203             FaceContourGraphic faceGraphic = new FaceContourGraphic(mGraphicOverlay);
204             mGraphicOverlay.add(faceGraphic);
205             faceGraphic.updateFace(face);
206         }
207
208         private void showToast(String message) {
209             Toast.makeText(getApplicationContext(), message, Toast.LENGTH_SHORT).show();
210         }
211
212         // Functions for loading images from app assets.
213
214         // Returns max image width, always for portrait mode. Caller needs to swap width / height for dp
215         // landscape mode.
216         private Integer getImageMaxWidth() {
217             if (mImageMaxWidth == null) {
218                 // Calculate the max width in portrait mode. This is done lazily.
219                 // wait for

```

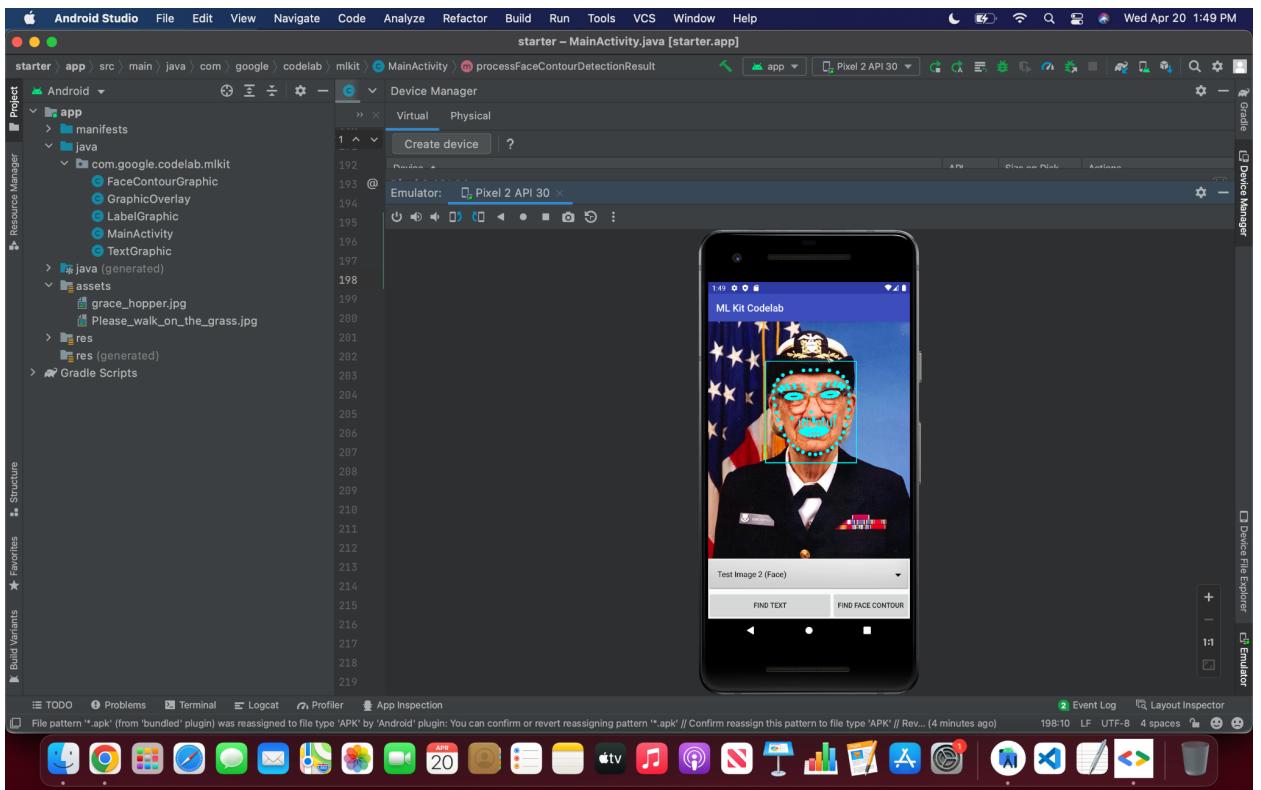
Project update recommended: Android Gradle Plugin can be upgraded.

Check the emulator



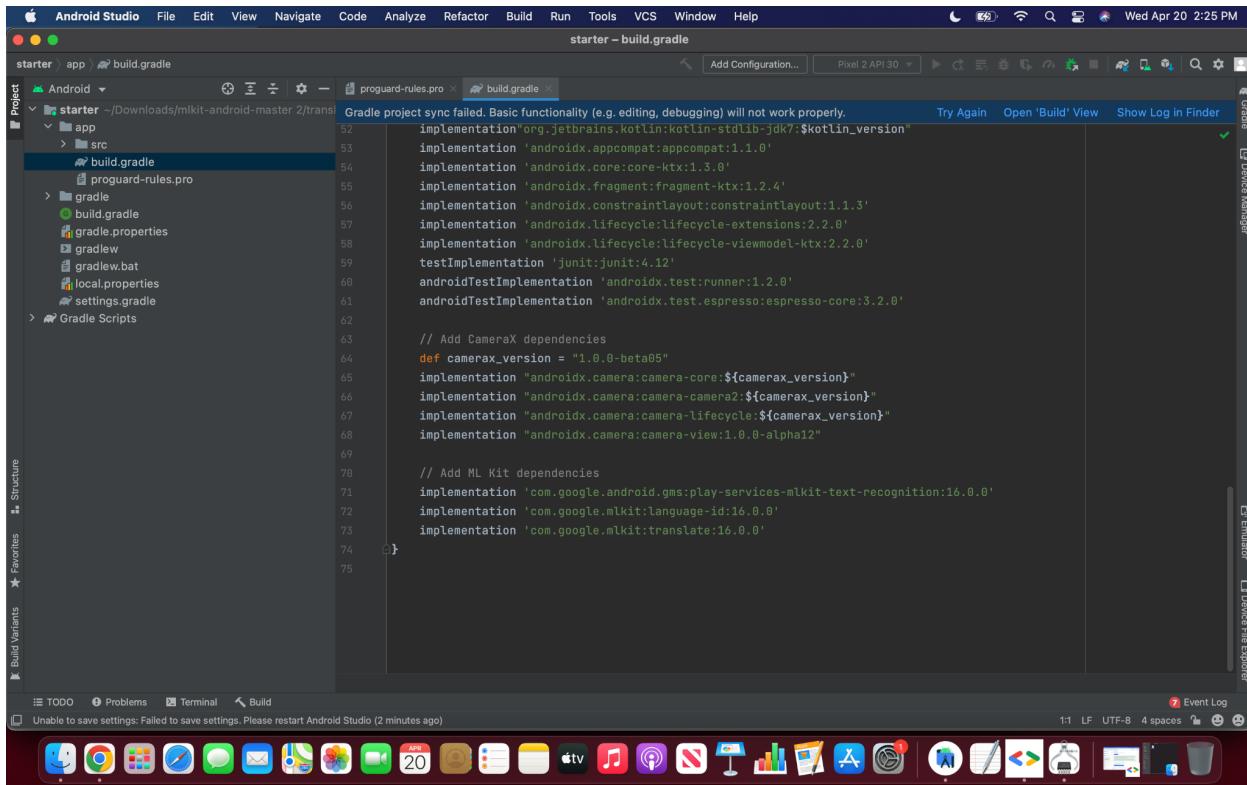
The screenshot shows the Android Studio interface with the emulator running. The emulator window displays a portrait of a person in a military uniform. At the top of the phone screen, it says "ML Kit Codelab". Below the image, there is a text box containing "Test Image 2 (Face)". At the bottom of the phone screen, there are two buttons: "FIND TEXT" and "FIND FACE CONTOUR". The Android Studio interface includes the Project, Device Manager, and other standard toolbars and panels.

- Find face contour



Recognize, Identify Language and Translate text with ML Kit and CameraX:Android

- Import project and check dependencies

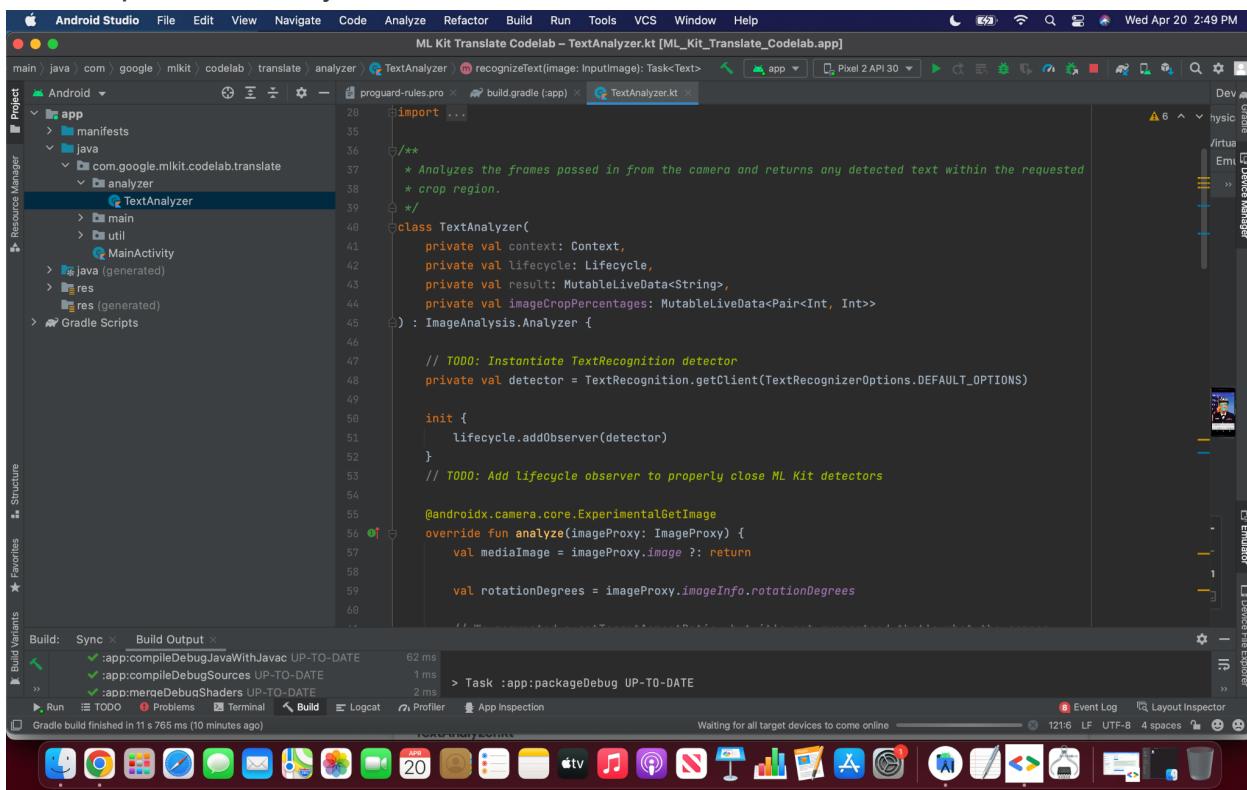


```
implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"
implementation 'androidx.appcompat:appcompat:1.1.0'
implementation 'androidx.core:core-ktx:1.3.0'
implementation 'androidx.fragment:fragment-ktx:1.2.4'
implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
implementation 'androidx.lifecycle:lifecycle-extensions:2.2.0'
implementation 'androidx.lifecycle:lifecycle-viewmodel-ktx:2.2.0'
testImplementation 'junit:junit:4.12'
androidTestImplementation 'androidx.test:runner:1.2.0'
androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0'

// Add CameraX dependencies
def camerax_version = "1.0.0-beta05"
implementation "androidx.camera:camera-core:${camerax_version}"
implementation "androidx.camera:camera-camera2:${camerax_version}"
implementation "androidx.camera:camera-lifecycle:${camerax_version}"
implementation "androidx.camera:camera-view:1.0.0-alpha12"

// Add ML Kit dependencies
implementation 'com.google.android.gms:play-services-mlkit-text-recognition:16.0.0'
implementation 'com.google.mlkit:language-id:16.0.0'
implementation 'com.google.mlkit:translate:16.0.0'
```

● Update textanalyzer.kt



```
import ...

/*
 * Analyzes the frames passed in from the camera and returns any detected text within the requested
 * crop region.
 */

class TextAnalyzer(
    private val context: Context,
    private val lifecycle: Lifecycle,
    private val result: MutableLiveData<String>,
    private val imageCropPercentages: MutableLiveData<Pair<Int, Int>>
) : ImageAnalysis.Analyzer {

    // TODO: Instantiate TextRecognition detector
    private val detector = TextRecognition.getClient(TextRecognizerOptions.DEFAULT_OPTIONS)

    init {
        lifecycle.addObserver(detector)
    }

    // TODO: Add lifecycle observer to properly close ML Kit detectors

    @Androidx.camera.core.ExperimentalGetImage
    override fun analyze(imageProxy: ImageProxy) {
        val mediaImage = imageProxy.image ?: return

        val rotationDegrees = imageProxy.imageInfo.rotationDegrees
```

ML Kit Translate Codelab – TextAnalyzer.kt [ML_Kit_Translate_Codelab.app]

```
private fun recognizeText(
    image: InputImage
): Task<Text> {
    // Pass image to an ML Kit Vision API
    return detector.process(image)
        .addOnSuccessListener { text ->
            // Task completed successfully
            result.value = text.text
        }
        .addOnFailureListener { exception ->
            // Task failed with an exception
            Log.e(TAG, "Text recognition error", exception)
            val message = getErrorMessage(exception)
            message?.let {
                Toast.makeText(context, message, Toast.LENGTH_SHORT).show()
            }
        }
}

private fun getErrorMessage(exception: Exception): String? {
    val mlKitException = exception as? MLKitException ?: return exception.message
    return if (mlKitException.errorCode == MLKitException.UNAVAILABLE) {
        "#Waiting for text recognition model to be downloaded"
    } else exception.message
}
```

Build: Sync x Build Output x

- ✓ :app:compileDebugJavaWithJavac UP-TO-DATE
- ✓ :app:compileDebugSources UP-TO-DATE
- ✓ :app:mergeDebugShaders UP-TO-DATE

Run TODO Problems Terminal Build Logcat App Inspection

Gradle build finished in 11 s 765 ms (10 minutes ago)

ML Kit Translate Codelab – TextAnalyzer.kt [ML_Kit_Translate_Codelab.app]

```
else -> Pair(widthCropPercent / 100f, heightCropPercent / 100f)

cropRect.inset(
    (imageWidth * widthCrop / 2).toInt(),
    (imageHeight * heightCrop / 2).toInt()
)
val croppedBitmap =
    ImageUtils.rotateAndCrop(convertImageToBitmap, rotationDegrees, cropRect)

// TODO call recognizeText() once implemented

recognizeText(InputImage.fromBitmap(croppedBitmap, 0)).addOnCompleteListener {
    imageProxy.close()
}
```

private fun recognizeText(
 image: InputImage
): Task<Text> {
 // Pass image to an ML Kit Vision API
 return detector.process(image)
 .addOnSuccessListener { text ->
 // Task completed successfully
 result.value = text.text
 }
 .addOnFailureListener { exception ->

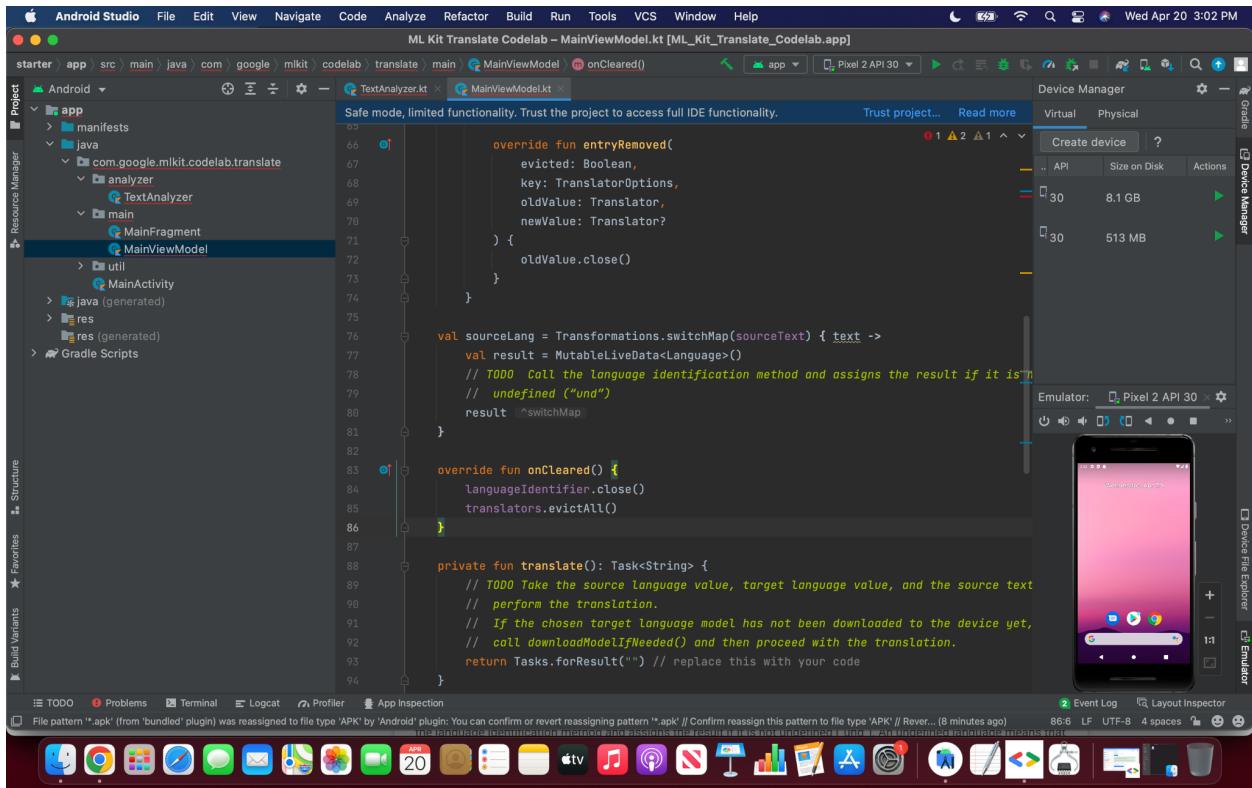
Build: Sync x Build Output x

- ✓ :app:compileDebugJavaWithJavac UP-TO-DATE
- ✓ :app:compileDebugSources UP-TO-DATE
- ✓ :app:mergeDebugShaders UP-TO-DATE

Run TODO Problems Terminal Build Logcat App Inspection

Gradle build finished in 11 s 765 ms (11 minutes ago)

- Update MainViewModel.kt



```

    override fun entryRemoved(
        evicted: Boolean,
        key: TranslatorOptions,
        oldValue: Translator,
        newValue: Translator?
    ) {
        oldValue.close()
    }

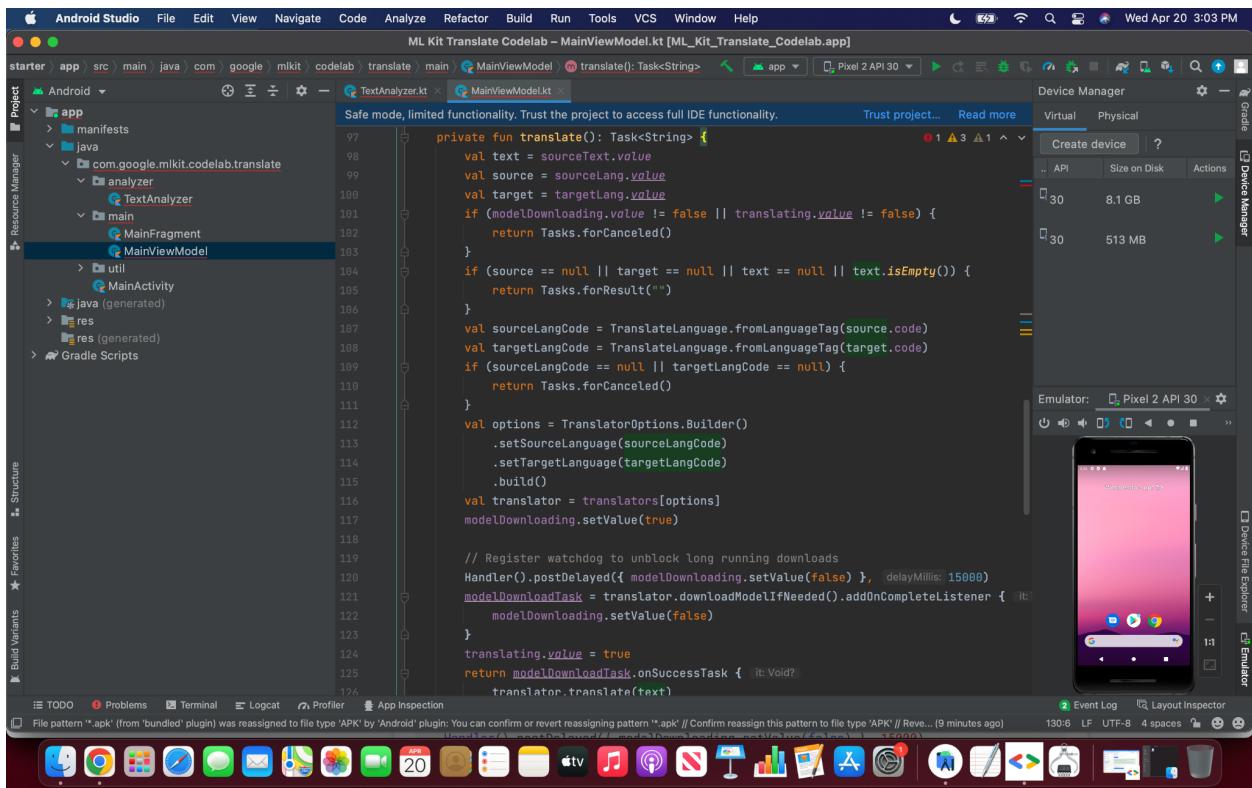
    val sourceLang = Transformations.switchMap(sourceText) { text ->
        val result = MutableLiveData<Language>()
        // TODO Call the language identification method and assigns the result if it is undefined ("und")
        result ^switchMap {
            ...
        }
        result
    }

    override fun onCleared() {
        languageIdentifier.close()
        translators.evictAll()
    }

    private fun translate(): Task<String> {
        // TODO Take the source language value, target language value, and the source text
        // perform the translation.
        // If the chosen target Language model has not been downloaded to the device yet,
        // call downloadModelIfNeeded() and then proceed with the translation.
        return Tasks.forResult("") // replace this with your code
    }
}

```

- Update translate method



```

private fun translate(): Task<String> {
    val text = sourceText.value
    val source = sourceLang.value
    val target = targetLang.value
    if (modelDownloading.value != false || translating.value != false) {
        return Tasks.forCancelled()
    }
    if (source == null || target == null || text == null || text.isEmpty()) {
        return Tasks.forResult("")
    }
    val sourceLangCode = TranslateLanguage.fromLanguageTag(source.code)
    val targetLangCode = TranslateLanguage.fromLanguageTag(target.code)
    if (sourceLangCode == null || targetLangCode == null) {
        return Tasks.forCancelled()
    }
    val options = TranslatorOptions.Builder()
        .setSourceLanguage(sourceLangCode)
        .setTargetLanguage(targetLangCode)
        .build()
    val translator = translators[options]
    modelDownloading.setValue(true)

    // Register watchdog to unblock long running downloads
    Handler().postDelayed({ modelDownloading.setValue(false) }, delayMillis: 15000)
    modelDownloadTask = translator.downloadModelIfNeeded().addOnCompleteListener {
        modelDownloading.setValue(false)
    }
    translating.value = true
    return modelDownloadTask.onSuccessTask { it: Void? -
        translator.translate(text)
    }
}

```

- Application run

