TeachEngineering

Machine Learning for Kids Tutorial

Use the ML4K Tutorial adapted from Scratch 3 to recognize food instead of books.

< Back to project

What have you done?

You have collected examples of images for a computer to use to recognise when images are Low_Glucose_55_or_less, Medium_Glucose_56_to_69 or High_Glucose_70_and_more.

You've collected:

- 25 examples of Low_Glucose_55_or_less,
- 20 examples of Medium_Glucose_56_to_69,
- 26 examples of High_Glucose_70_and_more

What's next?

Ready to start the computer's training?

Click the button below to start training a machine learning model using the examples you have collected so far

(Or go back to the Train page if you want to collect some more examples first.)

Info from training computer:

Train new machine learning model

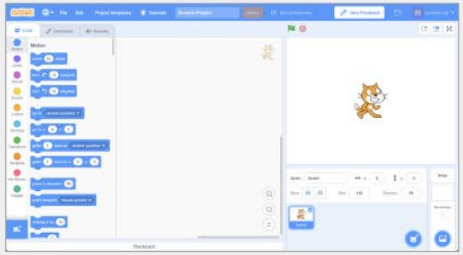
Make something with your machine learning model

< Back to project

Scratch 3

Use your machine learning model in Scratch

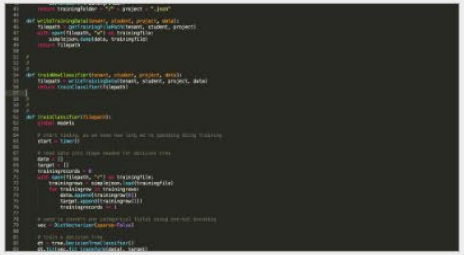
Scratch 3



Python

Write Python code to use your machine learning model

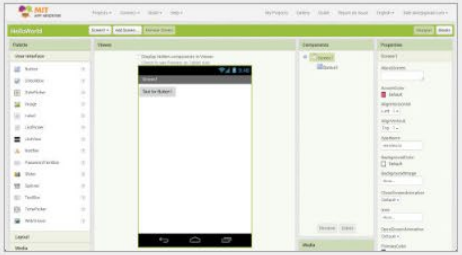
Python



App Inventor

Make a mobile app for your phone or tablet

App Inventor



Open in Scratch 3

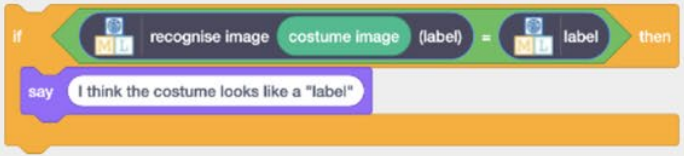
Your project will add these blocks to Scratch.

Put images in the input for this, and it will return the label that your machine learning model recognises it as.

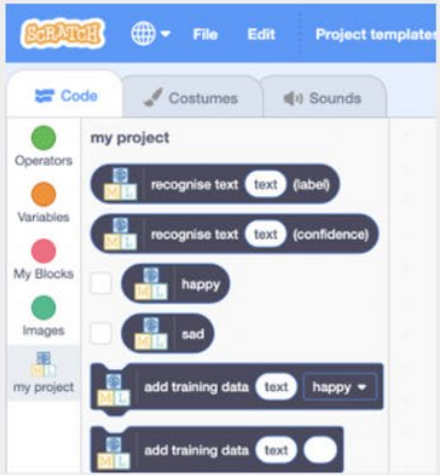
This will return how confident your machine learning model is that it recognises the type of images. (As a number from 0 - 100).

These blocks represent the labels you've created in your project, so you can use their names in your scripts.

This means you can do something like this:

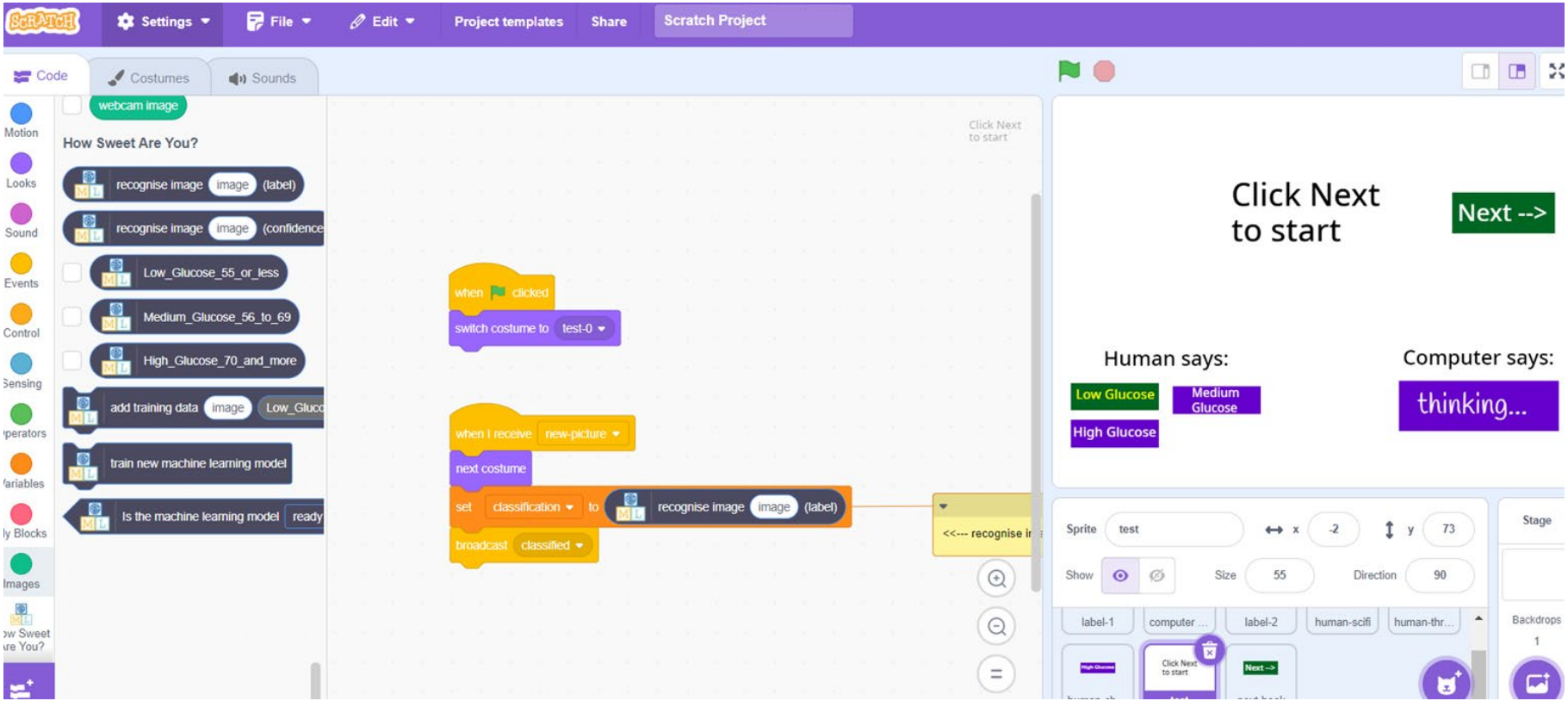


It will look something like this - except with the name of your project.

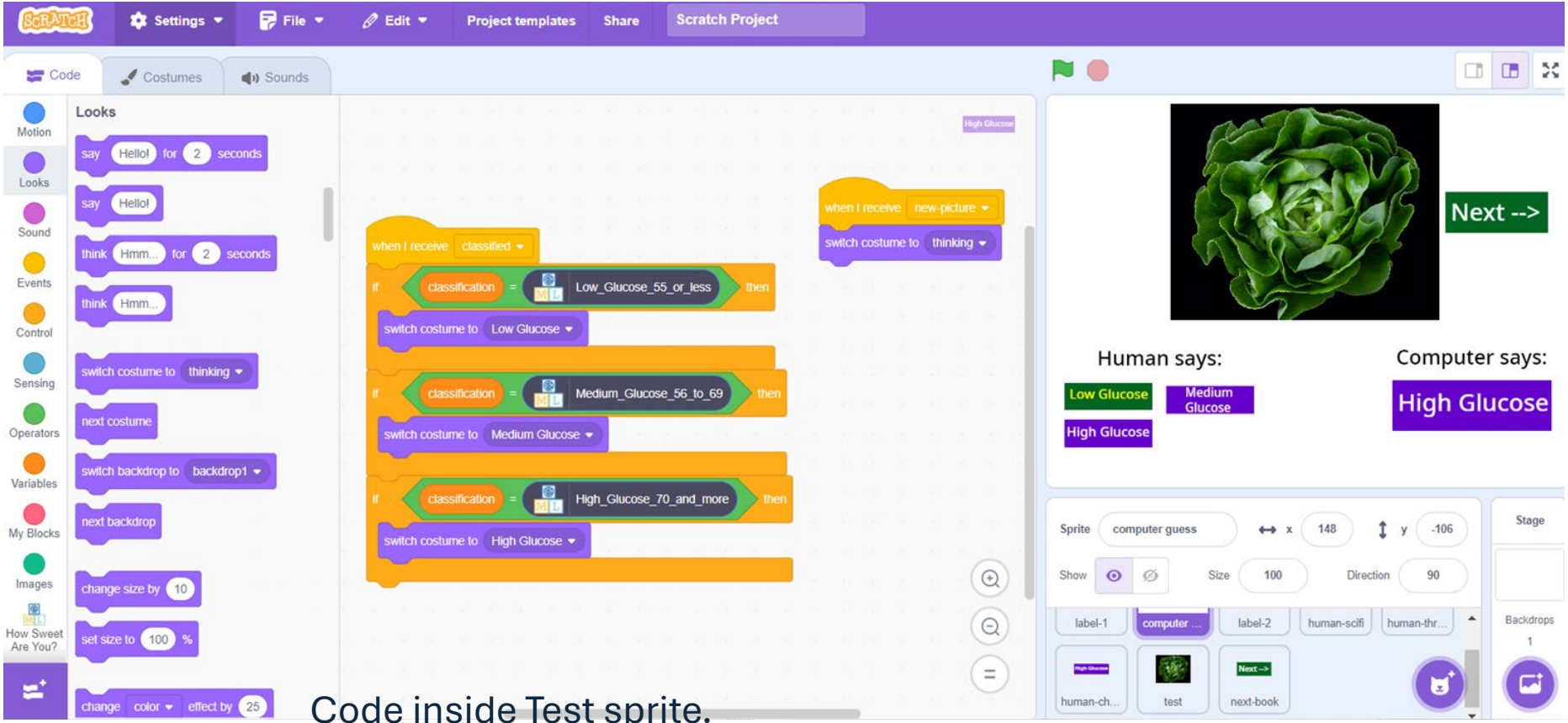


TeachEngineering

ncwit.org

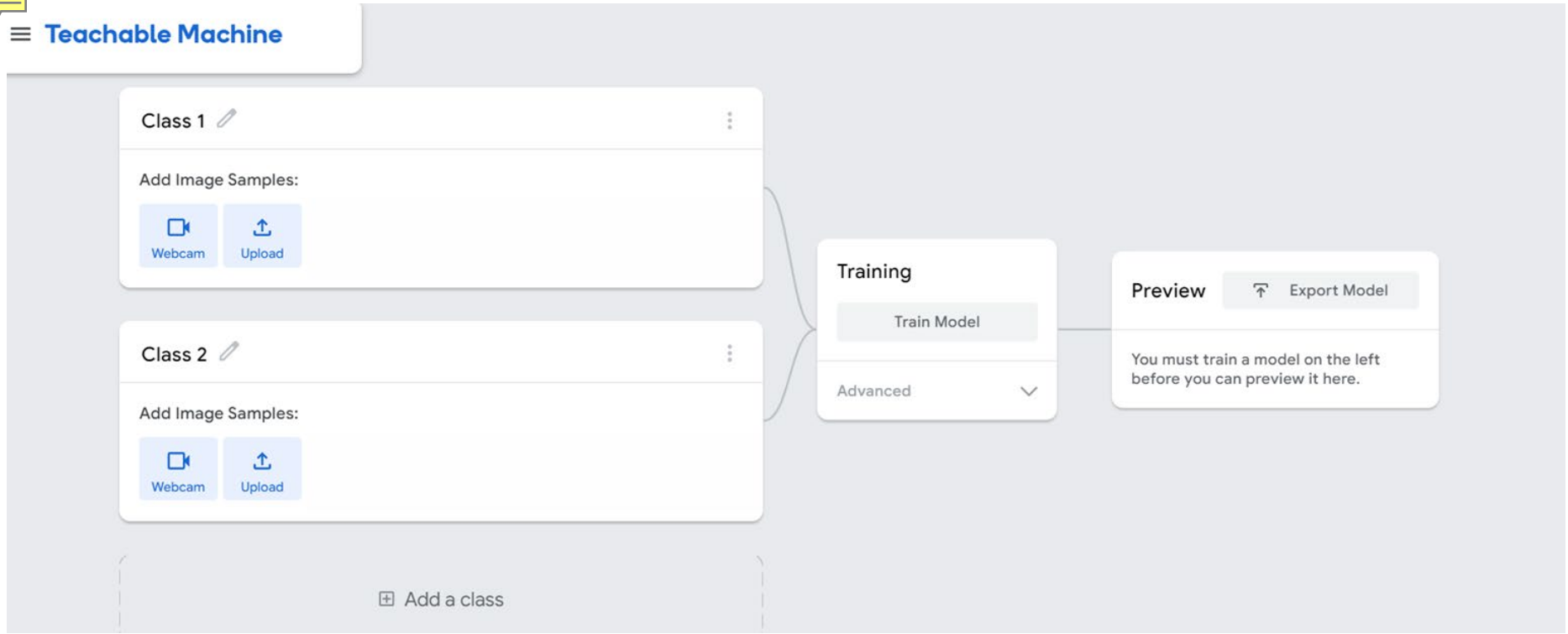


In Test sprite, add "new picture" and "recognize image."



Code inside Test sprite.

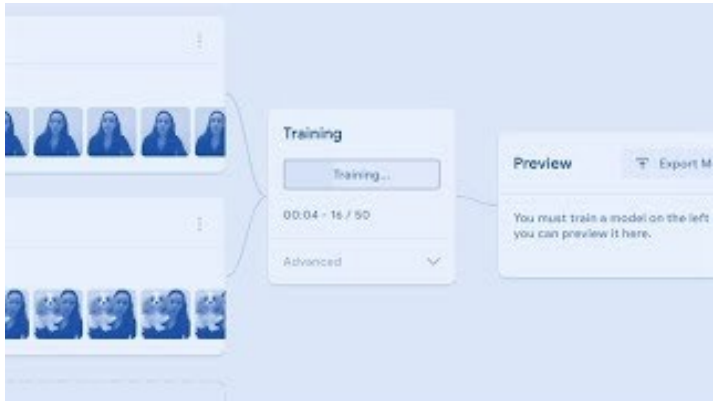
Teachable Machine Tutorial



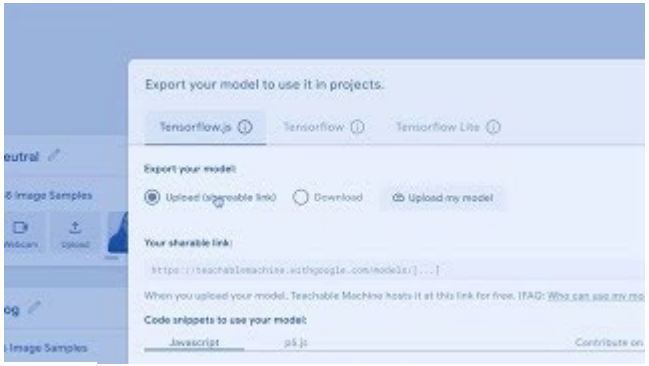
https://teachablemachine.withgoogle.com/train/image

Step 1: Gather data

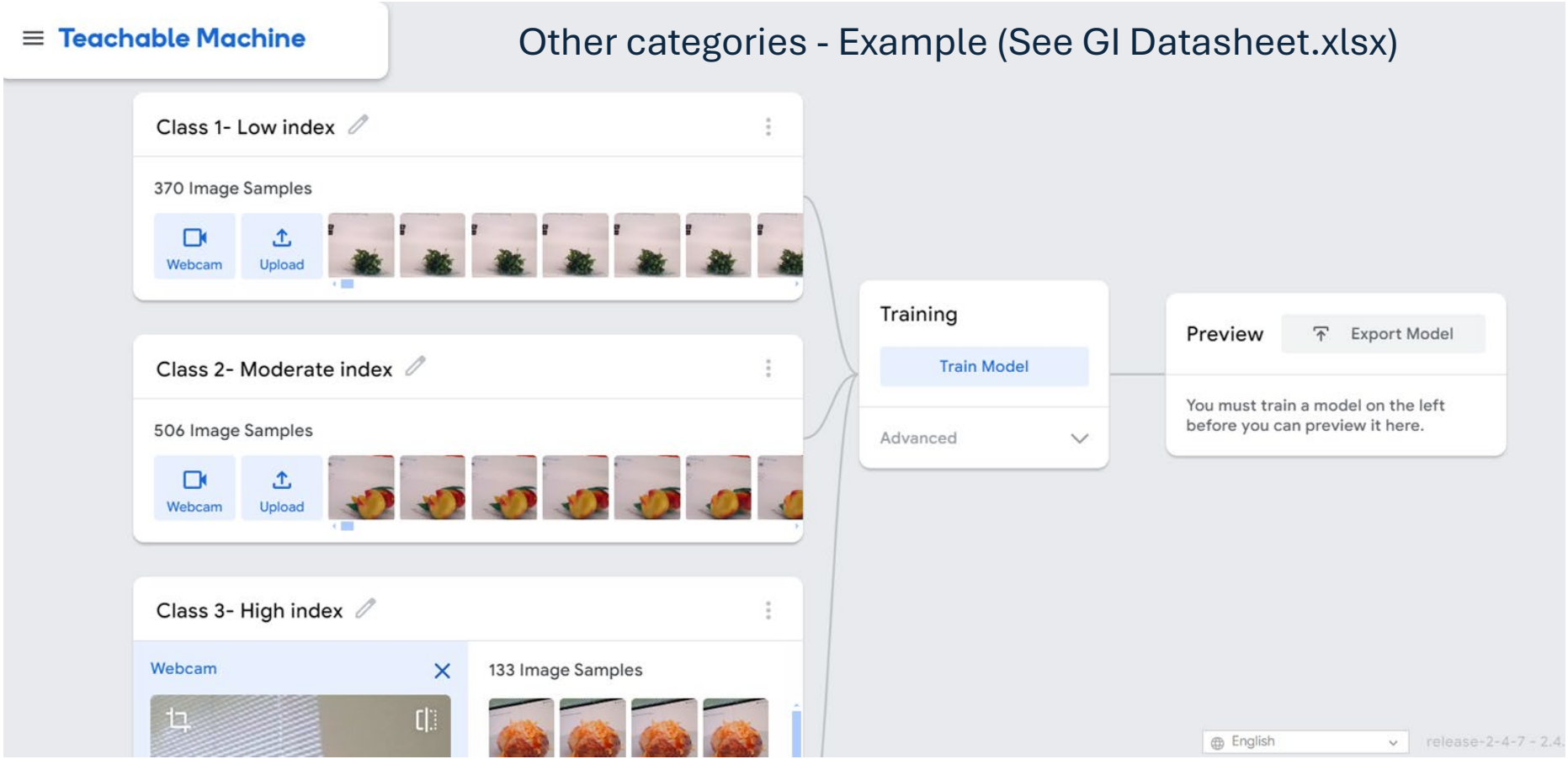
Step 2: Train the model

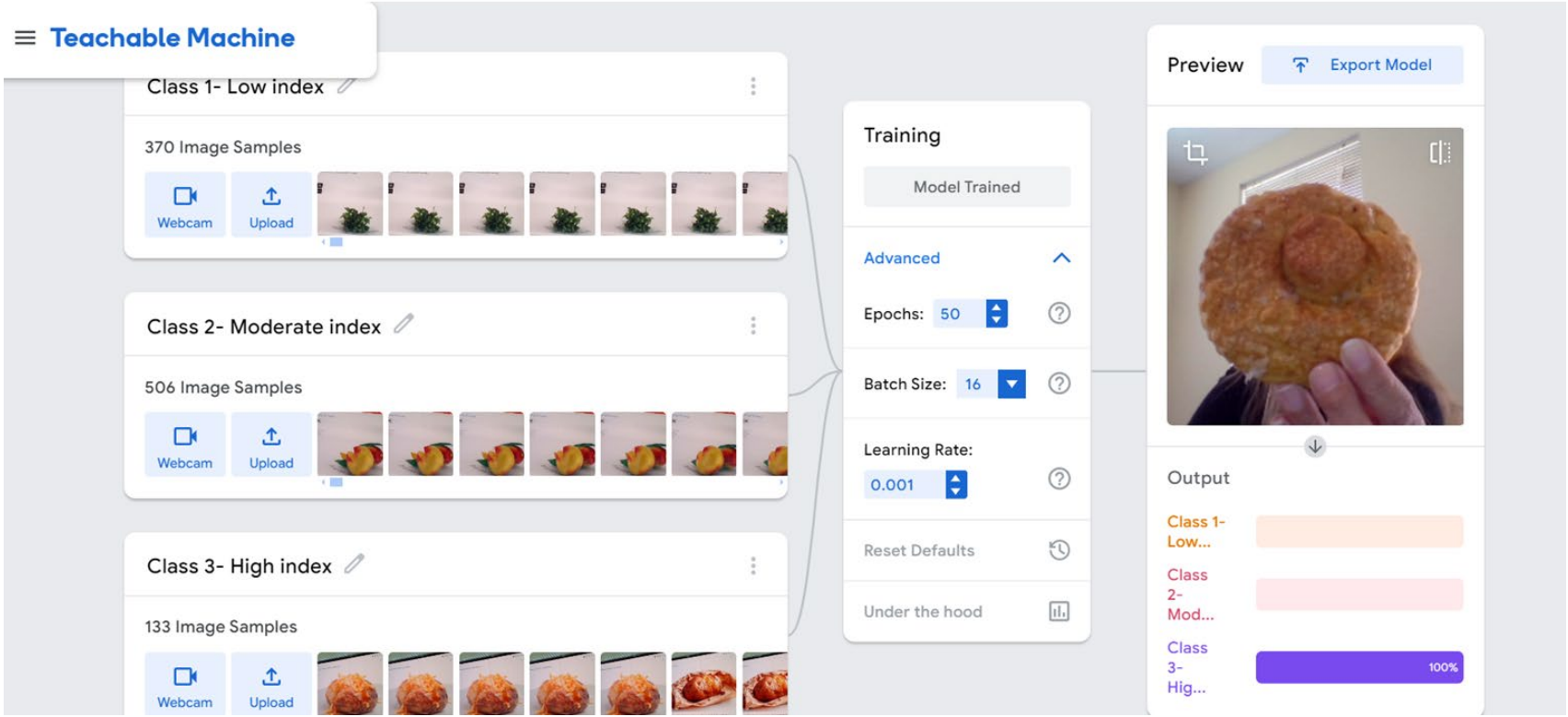


Step 3: Export



Other categories - Example (See GI Datasheet.xlsx)





I used 3D images online to build the database. I showed TM real food and was able to visually categorize it!

This is the Teachable Machine in action!

Teachable Machine splits your samples into two buckets. That's why you'll see two labels, training and test, in the graphs below.

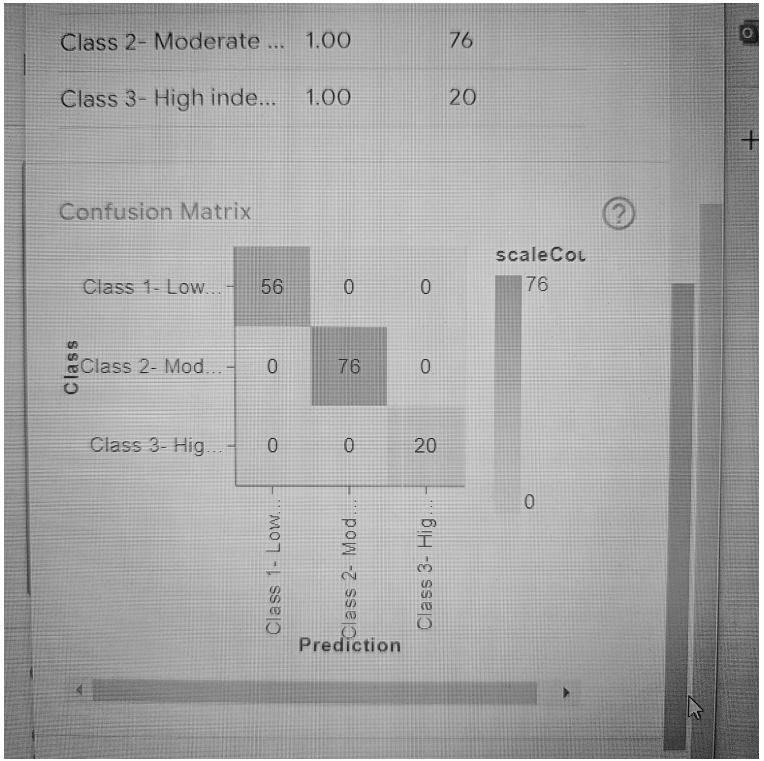
Training samples: (85% of the samples) are used to train the model how to correctly classify new samples into the classes you've made.

Test samples: (15% of the samples) are never used to train the model, so after the model has been trained on the training samples, they are used to check how well the model is performing on new, never-before-seen data.

Underfit: a model is underfit when it classifies poorly because the model hasn't captured the complexity of the training samples.

Overfit: a model is overfit when it learns to classify the training samples so closely that it fails to make correct classifications on the test samples.

Epochs: One epoch means that every training sample has been fed through the model at least once. If your epochs are set to 50, for example, it means that the model you are training will work through the entire training dataset 50 times.



```
<div>Teachable Machine Image Model</div>
<button type="button" onclick="init()">Start</button>
<div id="webcam-container"></div>
<div id="label-container"></div>
<script
src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@latest/dist/tf.min.js"></script>
<script
src="https://cdn.jsdelivr.net/npm/@teachablemachine/image@latest/dist/teachablema
chine-image.min.js"></script>
<script type="text/javascript">
  // More API functions here:
  // https://github.com/googlecreativelab/teachablemachine-
community/tree/master/libraries/image

  // the link to your model provided by Teachable Machine export panel
  const URL = "./my_model/";

  let model, webcam, labelContainer, maxPredictions;

  // Load the image model and setup the webcam
  async function init() {
    const modelURL = URL + "model.json";
    const metadataURL = URL + "metadata.json";

    // load the model and metadata
    // Refer to tmImage.loadFromFiles() in the API to support files from a file picker
    // or files from your local hard drive
    // Note: the pose library adds "tmImage" object to your window (window.tmImage)
    model = await tmImage.load(modelURL, metadataURL);
    maxPredictions = model.getTotalClasses();

    // Convenience function to setup a webcam
    const flip = true; // whether to flip the webcam
    webcam = new tmImage.Webcam(200, 200, flip); // width, height, flip
    await webcam.setup(); // request access to the webcam
    await webcam.play();
    window.requestAnimationFrame(loop);

    // append elements to the DOM
    document.getElementById("webcam-
container").appendChild(webcam.canvas);
    labelContainer = document.getElementById("label-container");
    for (let i = 0; i < maxPredictions; i++) { // and class labels
      labelContainer.appendChild(document.createElement("div"));
    }
  }

  async function loop() {
    webcam.update(); // update the webcam frame
    await predict();
    window.requestAnimationFrame(loop);
  }

  // run the webcam image through the image model
  async function predict() {
    // predict can take in an image, video or canvas html element
    const prediction = await model.predict(webcam.canvas);
    for (let i = 0; i < maxPredictions; i++) {
      const classPrediction =
        prediction[i].className + ": " + prediction[i].probability.toFixed(2);
      labelContainer.childNodes[i].innerHTML = classPrediction;
    }
  }
</script>
```

Vocab

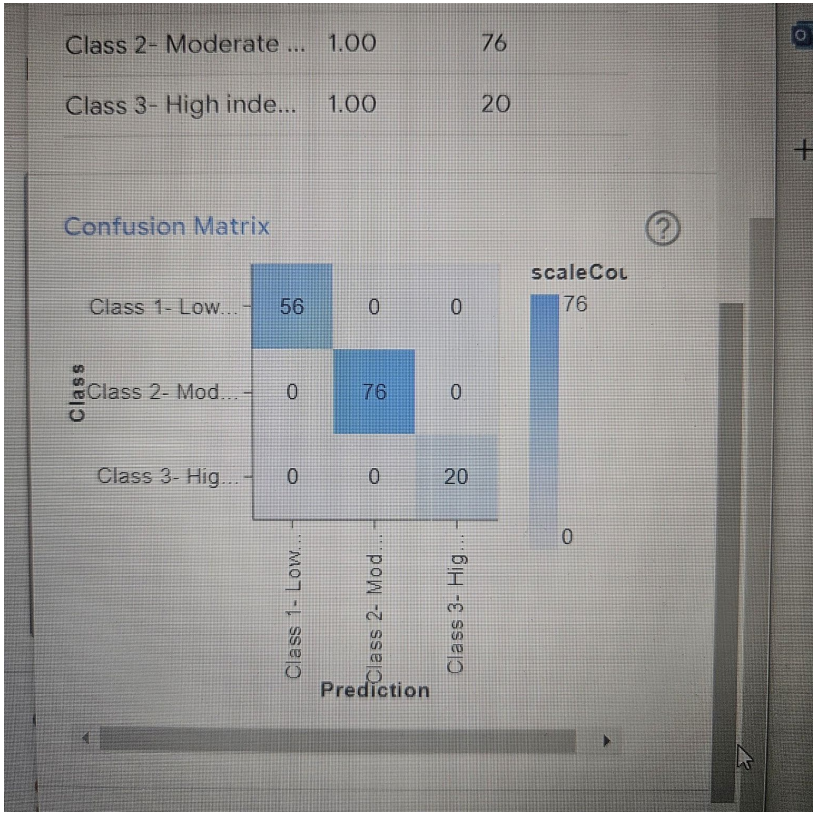
Accuracy per class

CLASS	ACCURACY	# SAMPLES
Class 1- Low index	1.00	56
Class 2- Moderate ...	1.00	76
Class 3- High inde...	1.00	20

Confusion Matrix

Calculate confusion matrix

Accuracy per epoch



Scan and use this model!

https://tinyurl.com/FoodModel