



# TeachEngineering

Machine Learning for Kids Tutorial



Subscribe to our newsletter at [TeachEngineering.org](https://TeachEngineering.org) to stay up-to-date on everything TE!

[ncwit.org](https://ncwit.org)

# 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

[About](#)[Teacher](#)[Projects](#)[Worksheets](#)[Pretrained](#)[Stories](#)[Book](#)[Help](#)[Log Out](#)

# Make something with your machine learning model

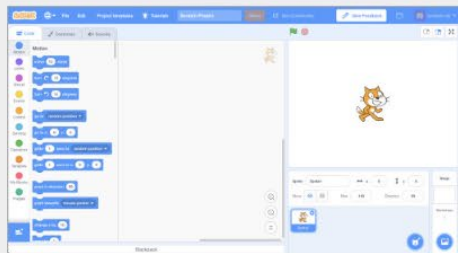
[← Back to project](#)

## Scratch 3

Use your machine learning model in Scratch



Scratch 3

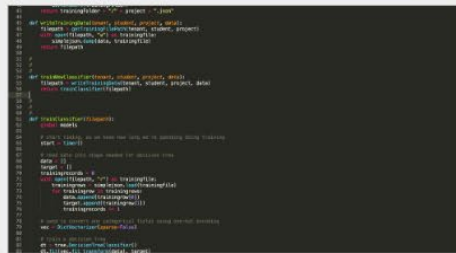


## Python

Write Python code to use your machine learning model



Python

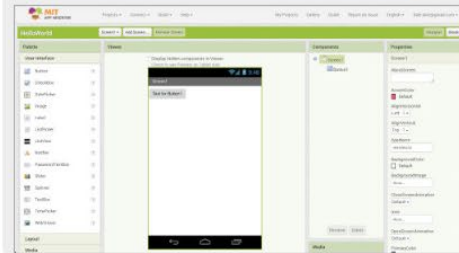


## App Inventor

Make a mobile app for your phone or tablet



App Inventor



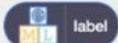
Your project will add these blocks to Scratch.



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

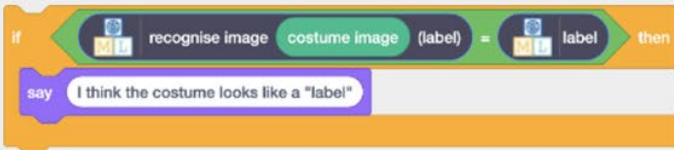


This will return how confident your machine learning model is that it recognises the type of images. (As a number from 0 - 100).

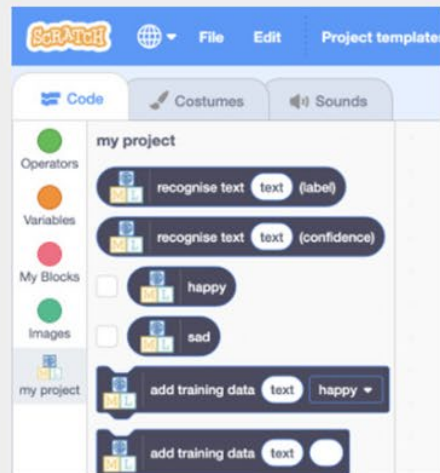


These blocks represent the labels you've created in your project, so you can use their names in your scripts.

**This means you can do something like this:**



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



The image shows a Scratch project titled "How Sweet Are You?". The interface includes a top menu bar with "Settings", "File", "Edit", "Project templates", "Share", and "Scratch Project". The left sidebar shows the "Code" tab with "Costumes" and "Sounds" sub-tabs. The "Code" tab is active, displaying a script area with the following blocks:

- when clicked** (Event)
- switch costume to test-0** (Costume)
- when I receive new-picture** (Message)
- next costume** (Costume)
- set classification to recognise image image (label)** (Sensing)
- broadcast classified** (Message)

The right side of the interface shows a test sprite named "test" with the following properties:

- Sprite:** test
- x:** -2
- y:** 73
- Size:** 55
- Direction:** 90

The bottom right corner shows a "Backdrops" panel with a list of backdrops: "label-1", "computer ...", "label-2", "human-scifi", and "human-thr...". The "Backdrops" panel also includes a "Click Next to start" button and a "Next -->" button.

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

Scratch Settings File Edit Project templates Share Scratch Project

Code Costumes Sounds

**Looks**

Motion

Looks

Sound

Events

Control

Sensing

Operators

Variables

My Blocks

Images

How Sweet Are You?

say Hello! for 2 seconds

say Hello!

think Hmm... for 2 seconds

think Hmm...

switch costume to thinking

next costume

switch backdrop to backdrop1

next backdrop

change size by 10

set size to 100 %

change color effect by 25

when I receive classified

if classification = Low\_Glucose\_55\_or\_less then

switch costume to Low Glucose

if classification = Medium\_Glucose\_56\_to\_69 then

switch costume to Medium Glucose

if classification = High\_Glucose\_70\_and\_more then

switch costume to High Glucose

when I receive new-picture

switch costume to thinking

High Glucose

Next -->

Human says:

Low Glucose Medium Glucose High Glucose

Computer says:

High Glucose

Sprite computer guess x 148 y -106

Show Size 100 Direction 90

label-1 computer ... label-2 human-scifi human-thr...



human-ch... test next-book

Backdrops 1


Code inside Test sprite.


# Teachable Machine Tutorial





Class 1  


Add Image Samples:


 Webcam


 Upload

Class 2  

Add Image Samples:

 Webcam

 Upload

 Add a class

### Training

Train Model

Advanced 

### Preview

 Export Model

You must train a model on the left before you can preview it here.

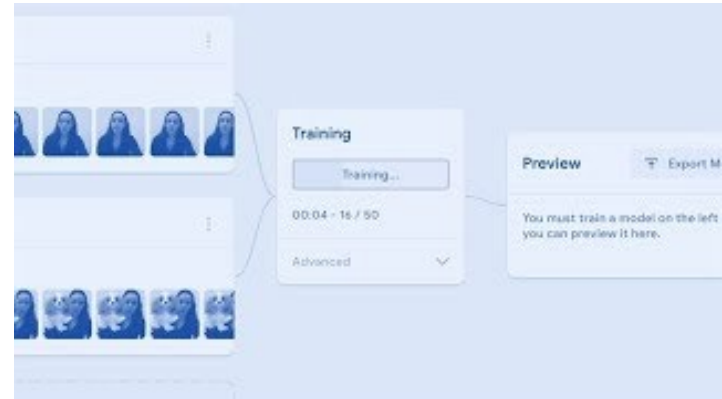
<https://teachablemachine.withgoogle.com/train/image>



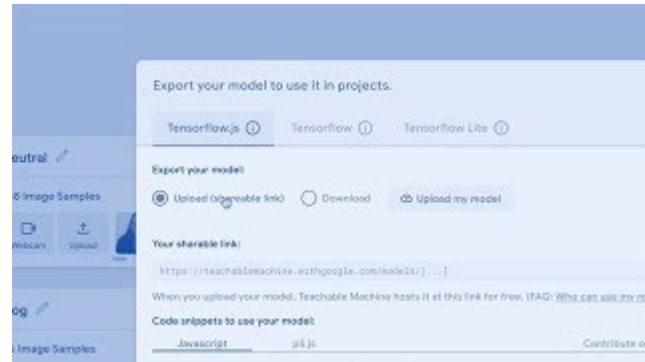
## Step 1: Gather data




## Step 2: Train the model



## Step 3: Export



## Other categories - Example (See GI Datasheet.xlsx)

Class 1- Low index 

370 Image Samples



Webcam



Upload



Class 2- Moderate index 

506 Image Samples



Webcam



Upload



Class 3- High index 

Webcam



133 Image Samples



Training

Train Model

Advanced



Preview



Export Model

You must train a model on the left before you can preview it here.

English

release-2-4-7 - 2.4.

## Teachable Machine

Class 1- Low index

370 Image Samples



Class 2- Moderate index

506 Image Samples



Class 3- High index

133 Image Samples



### Training

Model Trained

Advanced

Epochs: 50

Batch Size: 16

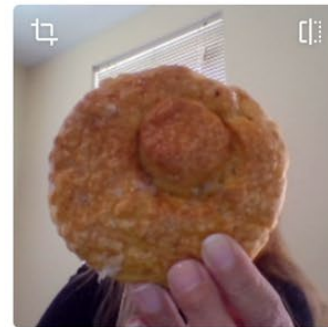
Learning Rate: 0.001

Reset Defaults

Under the hood

Preview

Export Model



Output

Class 1-  
Low...

Class  
2-  
Mod...

Class  
3-  
Hig...

100%

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.

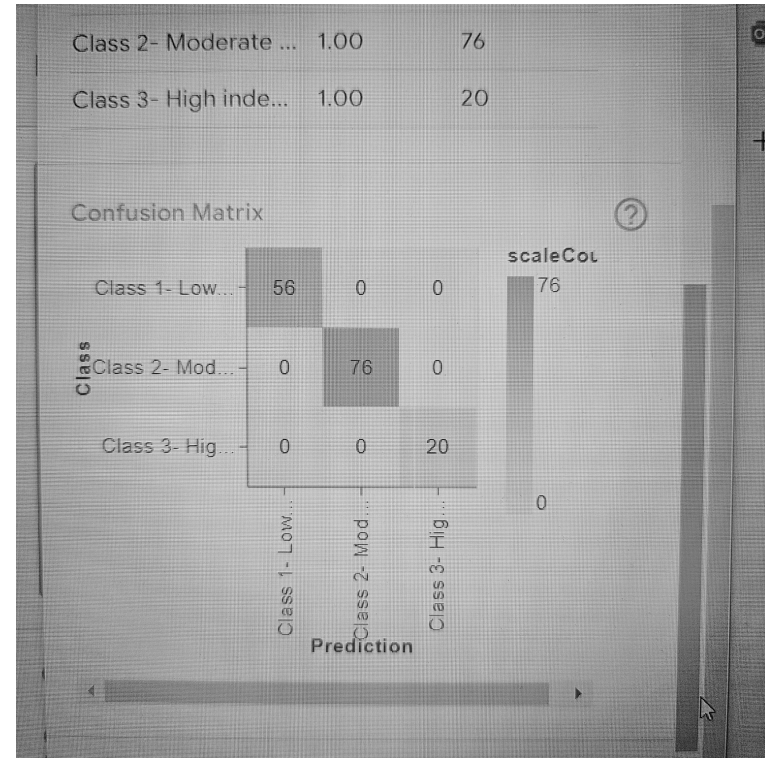
**Training samples:** (85% of the samples) are used to train the model how to correctly classify new samples into the classes you've made.

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

```

```

// Convenience function to setup a webcam
const flip = true; // whether to flip the webcam
webcam = new tmlImage.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 ^

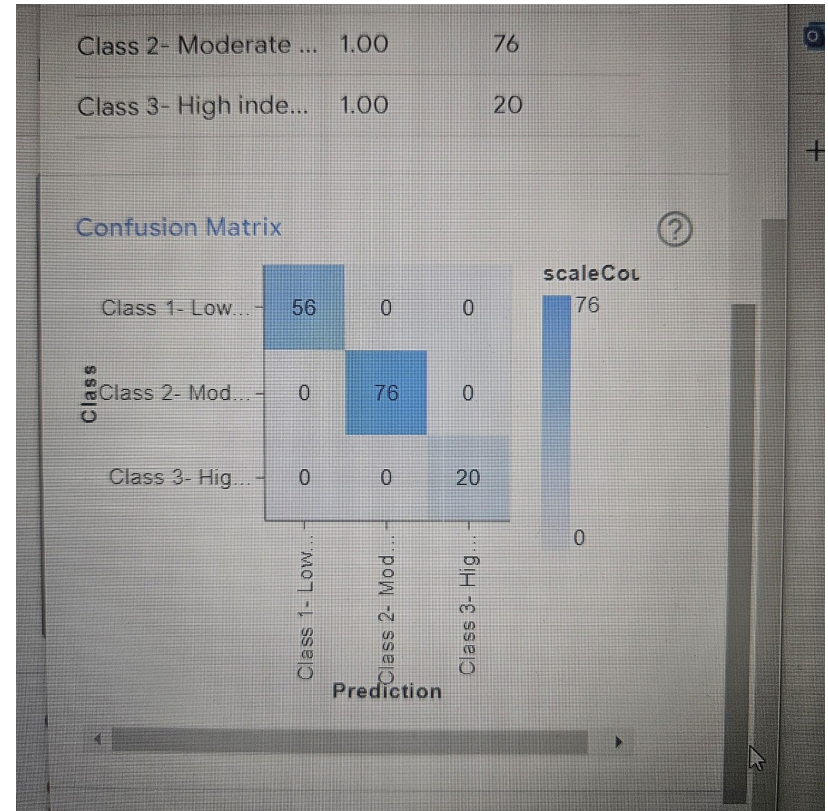
Accuracy per class ?

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>