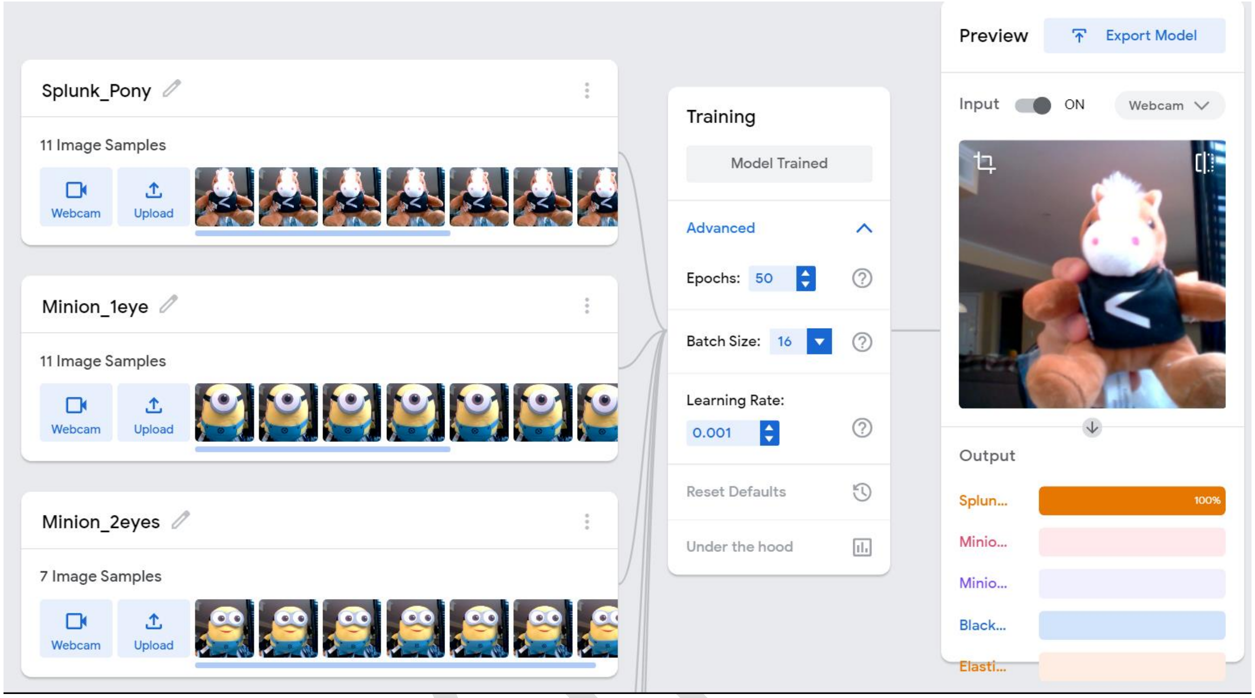
Google new Teachable Machine Learning demo in 10min ! (Easy Custom and Auto ML Models)



Epochs: 50

Batch Size: 16

Learning Rate: 0.001

Docs, Videos, Demos:

https://youtu.be/i9tjzr1KME0 by Dale Markowitz

Real time lab on your chrome browser:

https://teachablemachine.withgoogle.com/

Docs:

https://cloud.google.com/blog/products/ai-machine-learning/beginners-guide-to-painless-machine-learning – a lot of good info inside *** MUST READ MORE !

Videos:

https://youtu.be/n-zeeRLBgd0

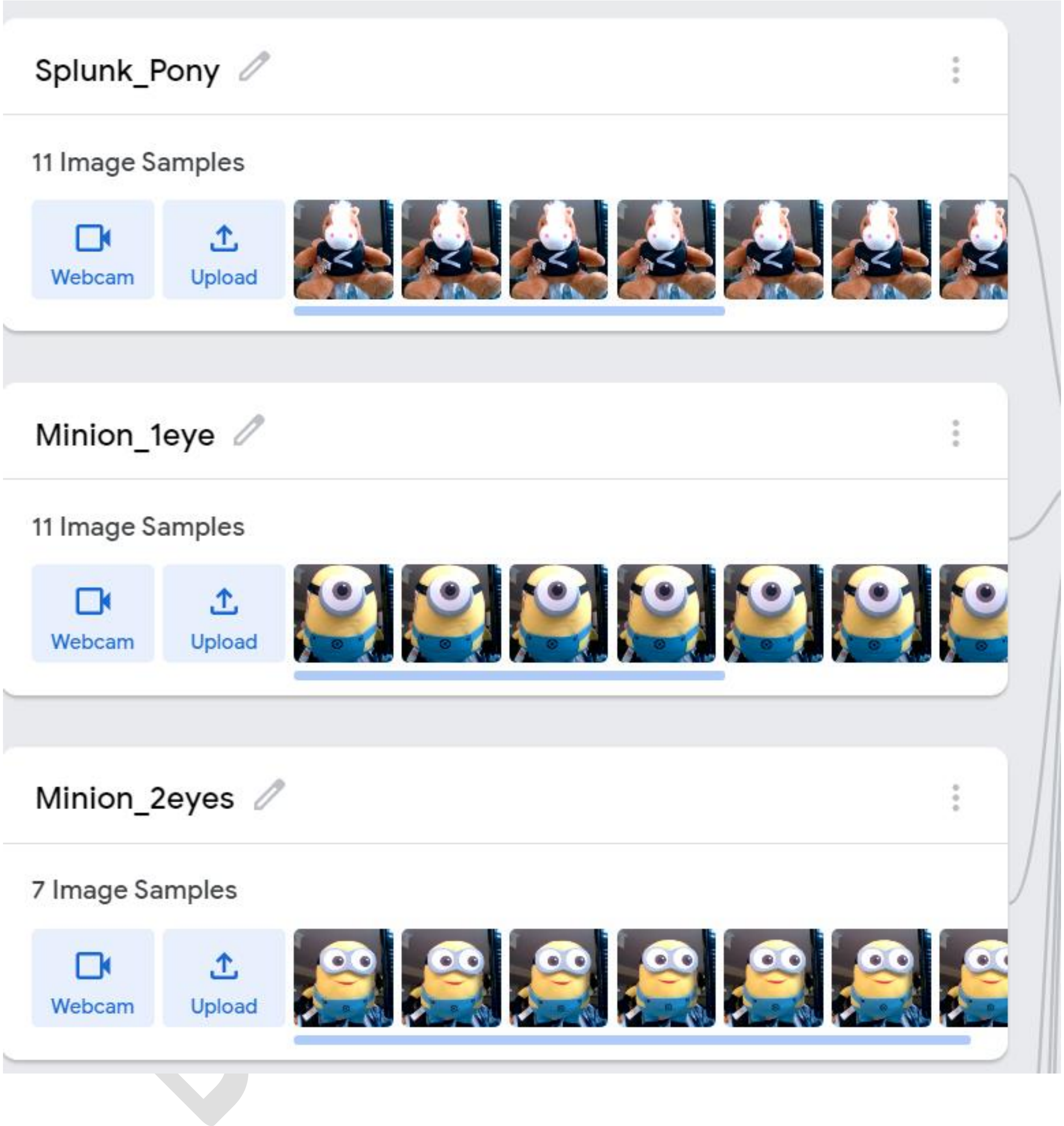
Export your model for your projects: sites, apps, and more. You can download your model or host it online for free. Learn more about Teachable Machine

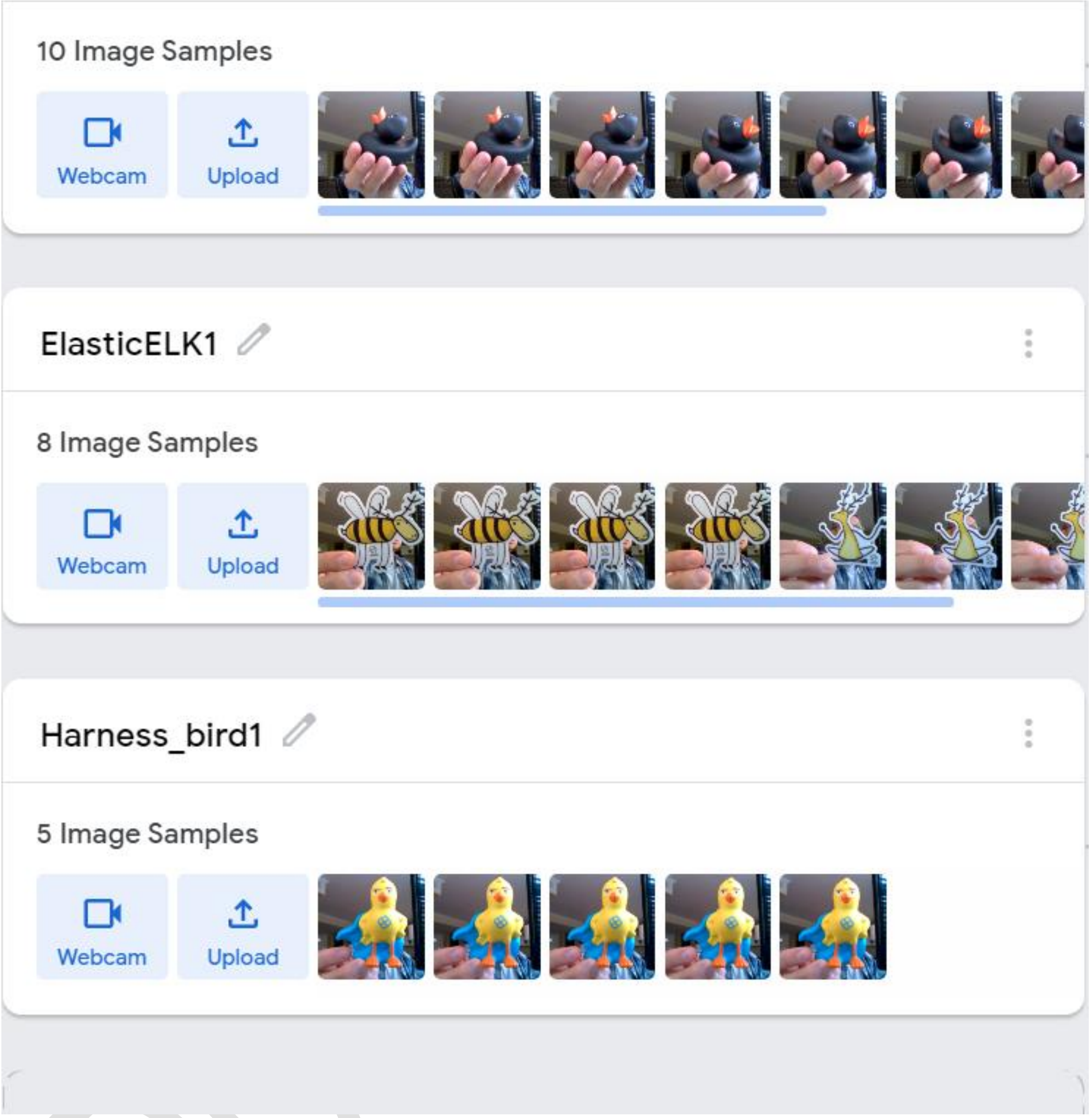
Git :

https://github.com/googlecreativelab/teachablemachine-community/tree/master/libraries/image

Step1: Prepare the Data, I used all the stuff animals and stickers from vendors: Splunk, Elastic, Harness, Blackduck, Minions, etc…

Easy to use the webcam (or photos) and just hold the button to take the number of images, e.g. side view, top view, etc. as many as you like:





10 Image Samples

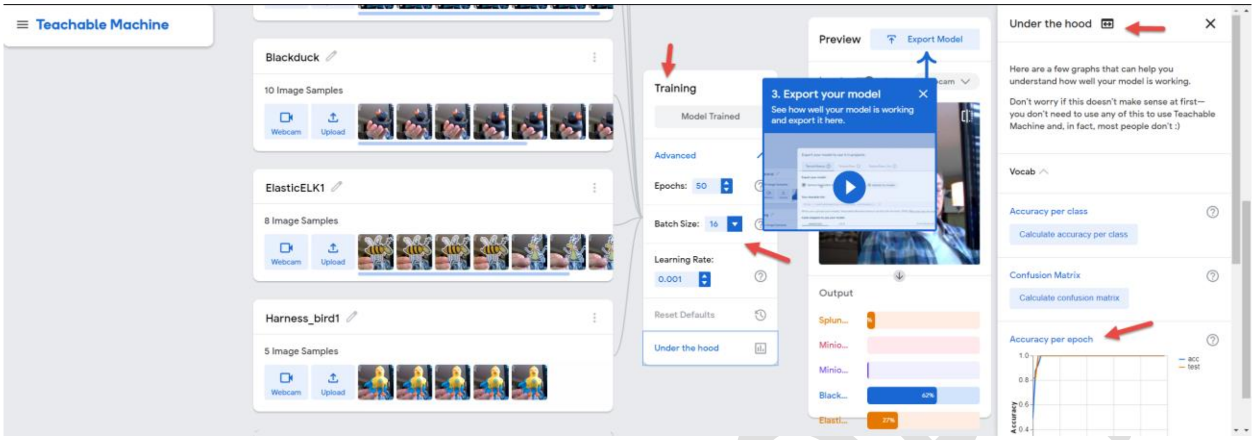
ElasticELK1

8 Image Samples

Harness_bird1

5 Image Samples

Step2: Train the model



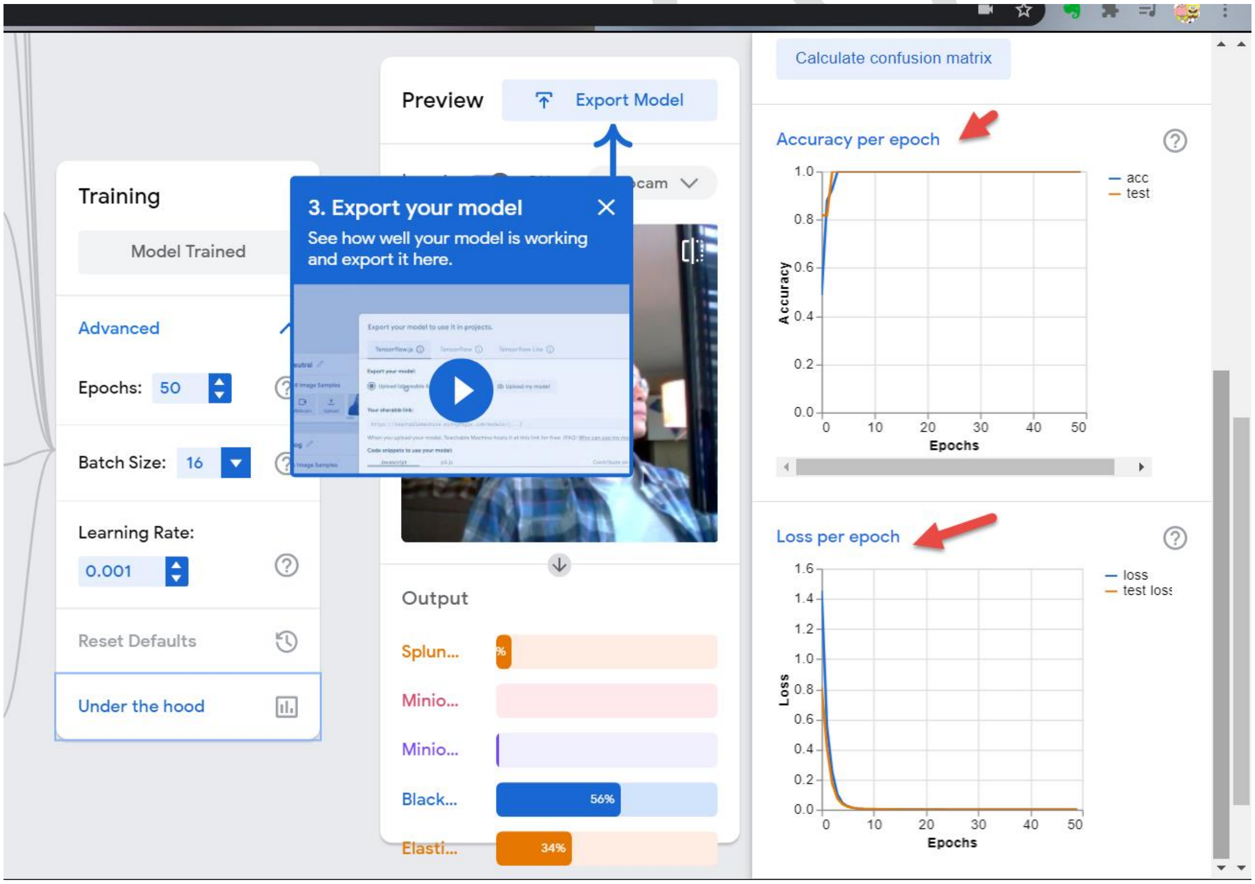
3. Export your model

See how well your model is working and export it here.

Here are a few graphs that can help you understand how well your model is working.

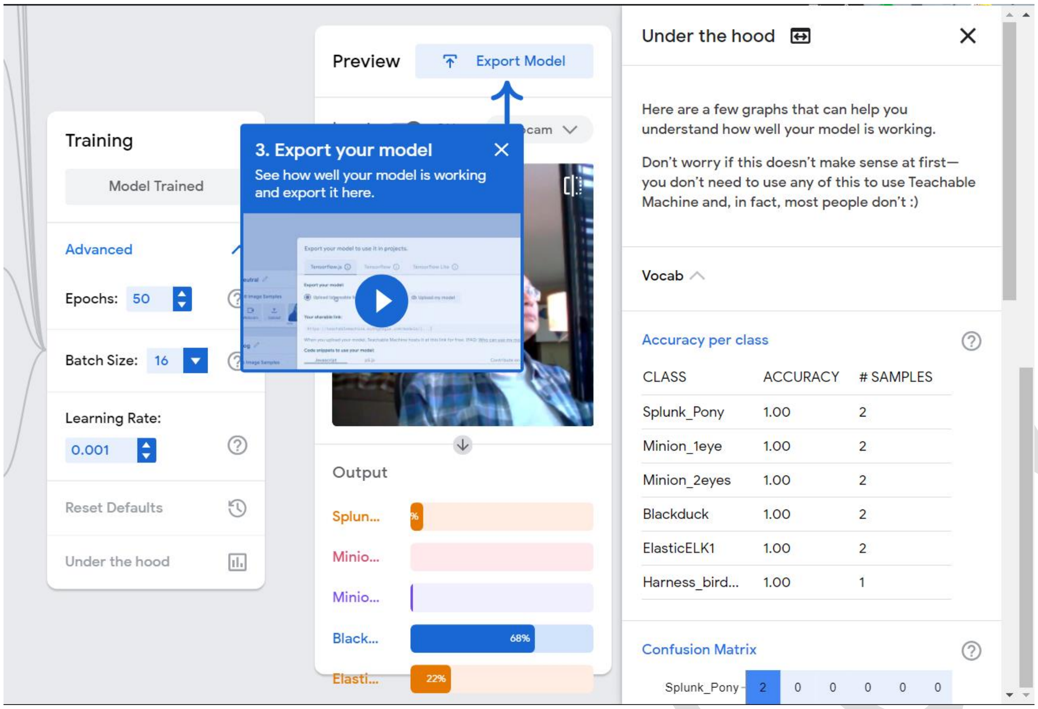
Don't worry if this doesn't make sense at first—you don't need to use any of this to use Teachable Machine and, in fact, most people don't :)

You can see a lot of details under the hood, e.g. epochs, learning rate, accuracy, confusion matrix, loss per epoch, etc…



3. Export your model

See how well your model is working and export it here.



Training

Model Trained

Advanced

Epochs: 50

Batch Size: 16

Learning Rate:

0.001

Reset Defaults

Under the hood

Preview

Export Model

3. Export your model

See how well your model is working and export it here.

Output

Splun...	%
Minio...	
Minio...	
Black...	68%
Elasti...	22%

Under the hood

Here are a few graphs that can help you understand how well your model is working.

Don't worry if this doesn't make sense at first—you don't need to use any of this to use Teachable Machine and, in fact, most people don't :)

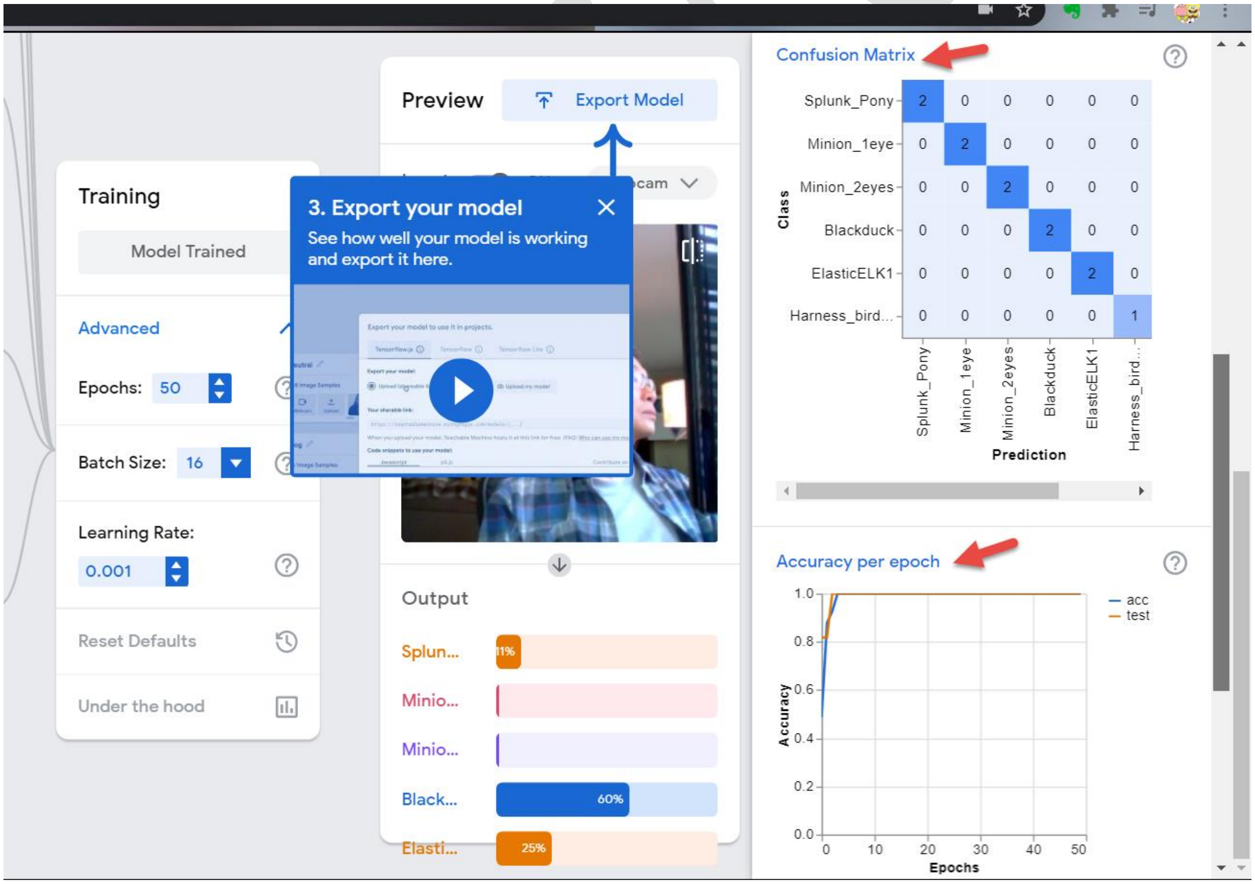
Vocab

Accuracy per class

CLASS	ACCURACY	# SAMPLES
Splunk_Pony	1.00	2
Minion_1eye	1.00	2
Minion_2eyes	1.00	2
Blackduck	1.00	2
ElasticELK1	1.00	2
Harness_bird...	1.00	1

Confusion Matrix

Splunk_Pony	2	0	0	0	0	0



Training

Model Trained

Advanced

Epochs: 50

Batch Size: 16

Learning Rate:

0.001

Reset Defaults

Under the hood

Preview

Export Model

3. Export your model

See how well your model is working and export it here.

Output

Splun...	11%
Minio...	
Minio...	
Black...	60%
Elasti...	25%

Confusion Matrix

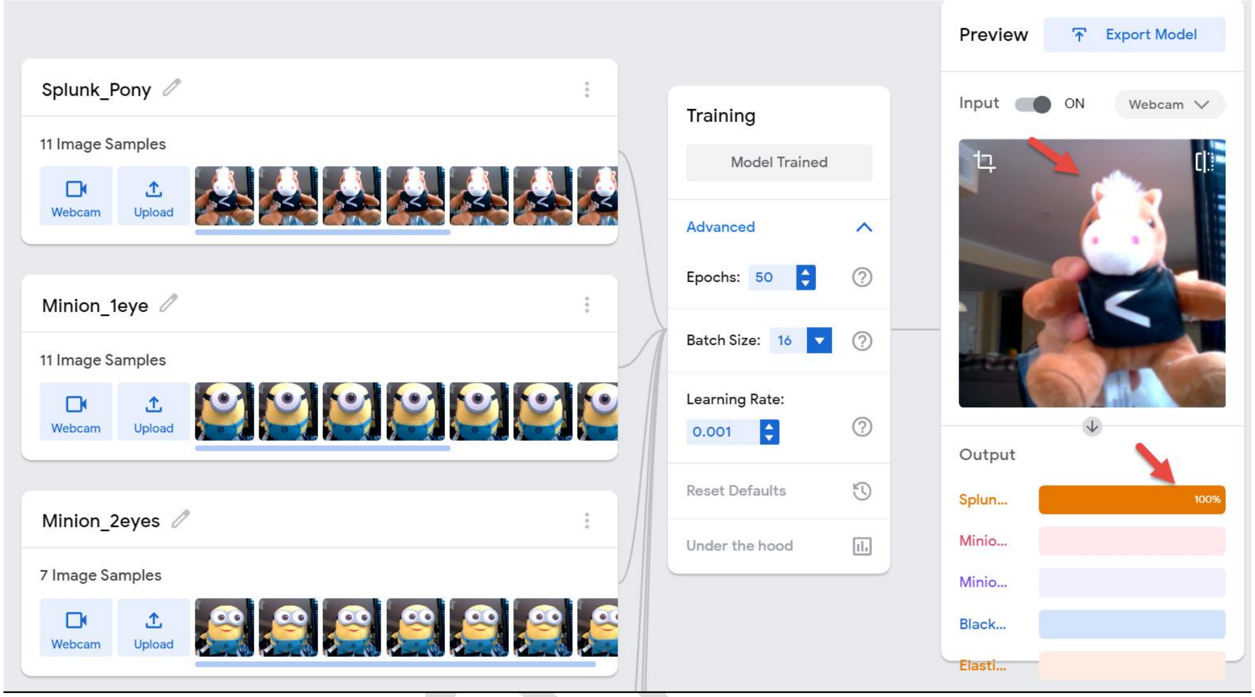
Class \ Prediction	Splunk_Pony	Minion_1eye	Minion_2eyes	Blackduck	ElasticELK1	Harness_bird...
Splunk_Pony	2	0	0	0	0	0
Minion_1eye	0	2	0	0	0	0
Minion_2eyes	0	0	2	0	0	0
Blackduck	0	0	0	2	0	0
ElasticELK1	0	0	0	0	2	0
Harness_bird...	0	0	0	0	0	1

Accuracy per epoch

Step 3: Test the model

You can use the real time webcam to test the image classification easily:

1/ 100% prediction for Splunk Pony

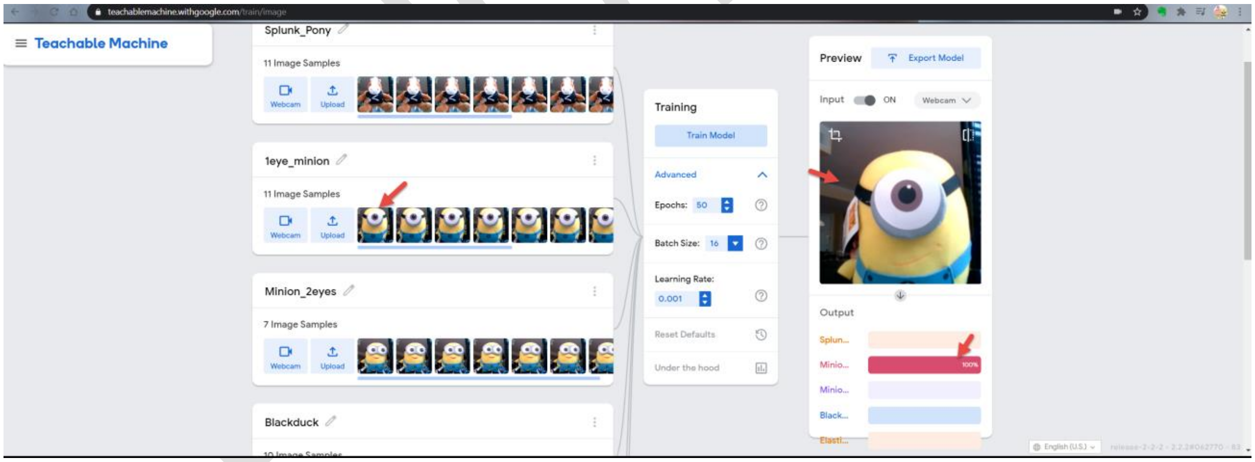


Epochs: 50

Batch Size: 16

Learning Rate: 0.001

2/ 100% good prediction for 1 eye minion

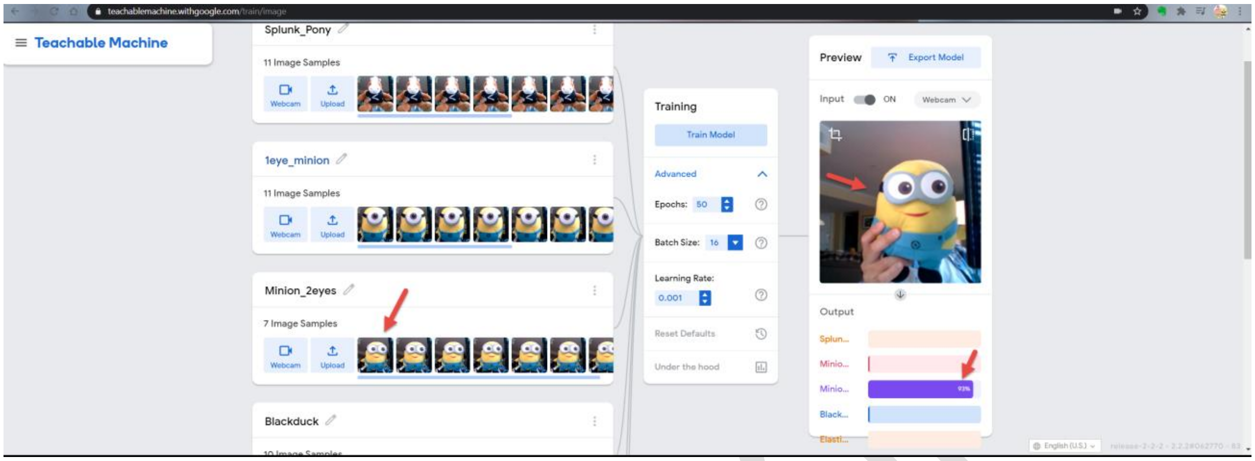


Epochs: 50

Batch Size: 16

Learning Rate: 0.001

3/ 2 eye minion a little confused with 1 eye minion above, so 93% prediction only

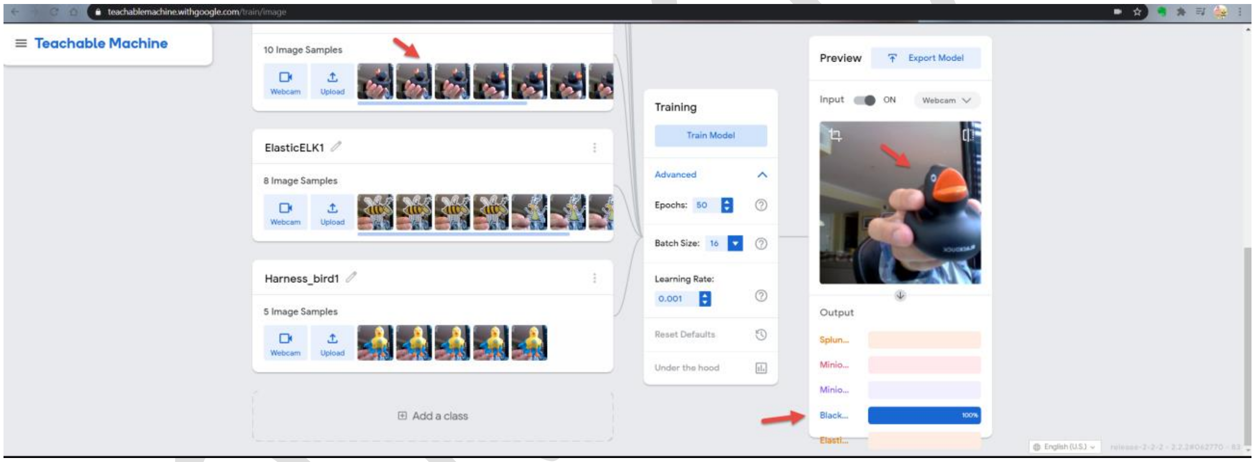


Epochs: 50

Batch Size: 16

Learning Rate: 0.001

4/ Blackduck 100% prediction

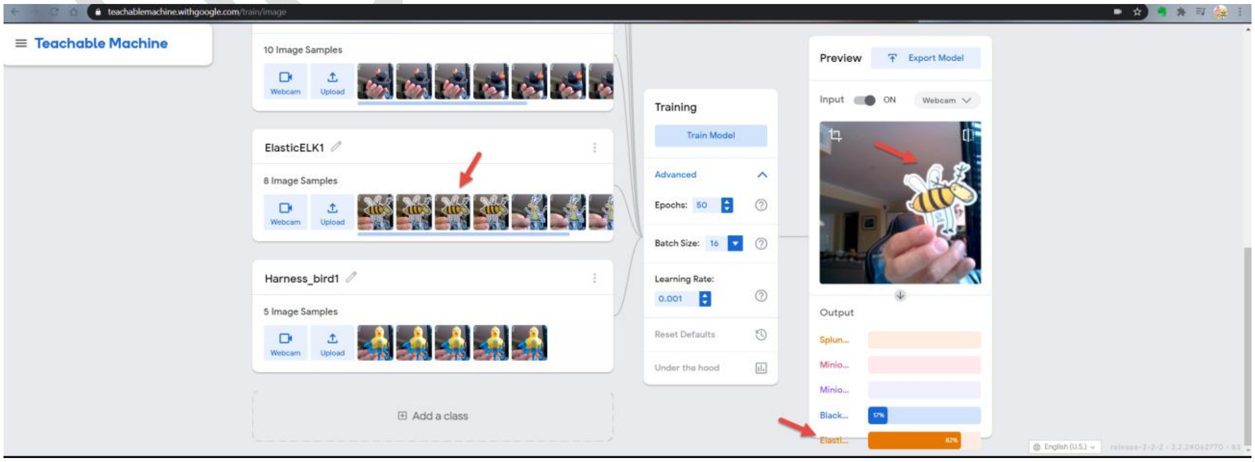


Epochs: 50

Batch Size: 16

Learning Rate: 0.001

5/ I moved the filebeat sticker far away from webcam, so elastic got only 82% prediction below

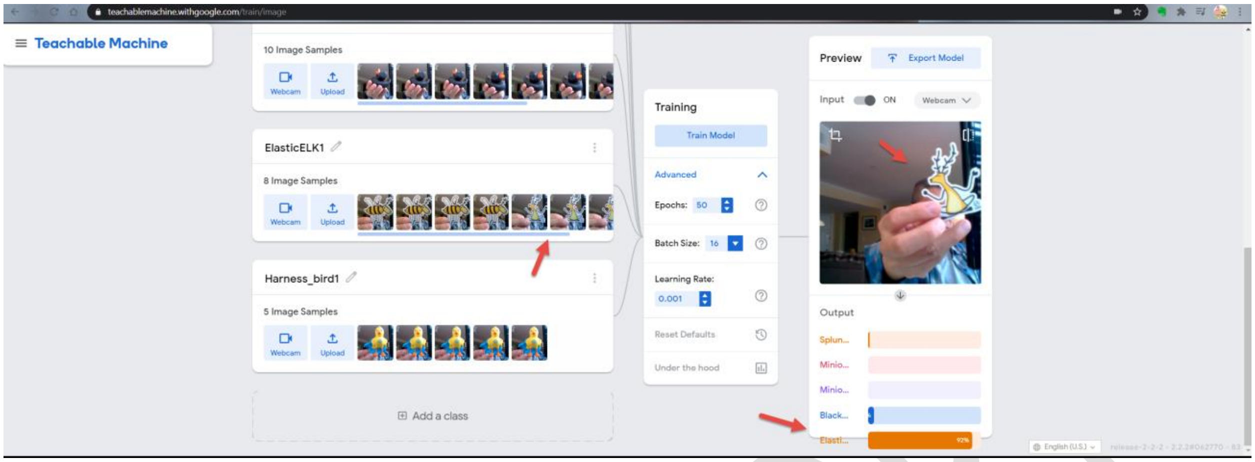


Epochs: 50

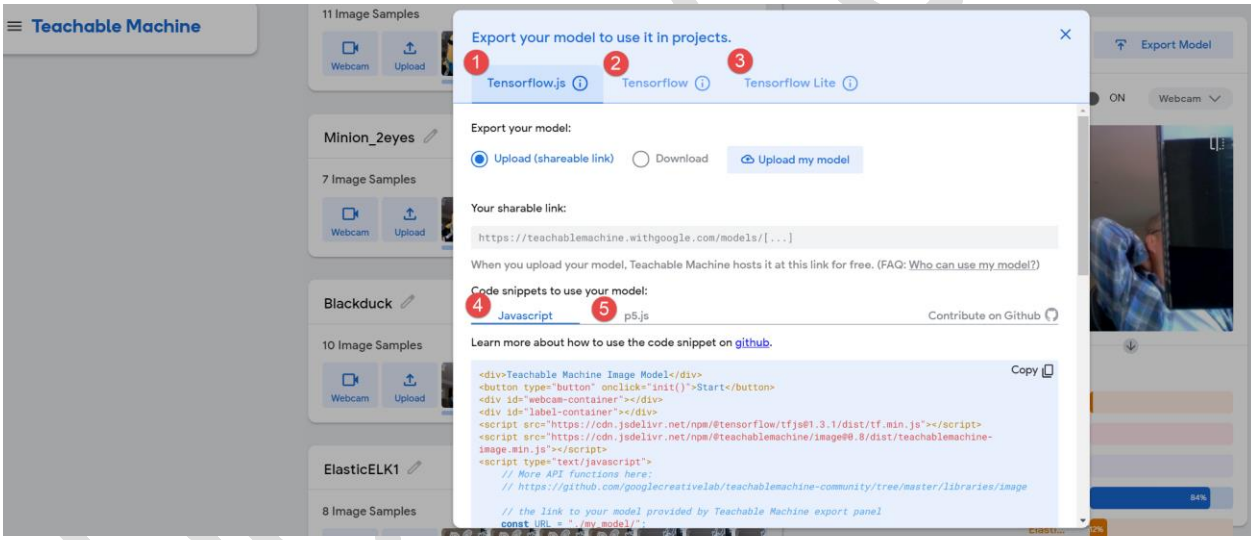
Batch Size: 16

Learning Rate: 0.001

6/ I moved the ELK sticker far away from webcam, so elastic ELK got only 92% prediction below



Step 4: Export and use the model



Export your model to use it in projects.

Export your model:

● Upload (shareable link) ○ Download Upload my model

Your sharable link:

https://teachablemachine.withgoogle.com/models/[...]

When you upload your model, Teachable Machine hosts it at this link for free. (FAQ: Who can use my model?)

Code snippets to use your model:

Javascript p5.js

Learn more about how to use the code snippet on github.

```
<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@1.3.1/dist/tf.min.js"></script>
<script src="https://cdn.jsdelivr.net/npm/@teachablemachine/image@0.8/dist/teachablemachine-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/";
```

1/ Tensorflow JS:

```
<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@1.3.1/dist/tf.min.js"></script>
```

```
<script src="https://cdn.jsdelivr.net/npm/@teachablemachine/image@0.8/dist/teachablemachine-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>
```

You can choose Tensorflow and Tensorflow Lite too.

2/ Tensorflow in Python:

```
import tensorflow.keras
from PIL import Image, ImageOps
import numpy as np

# Disable scientific notation for clarity
np.set_printoptions(suppress=True)

# Load the model
model = tensorflow.keras.models.load_model('keras_model.h5')

# Create the array of the right shape to feed into the keras model
# The 'length' or number of images you can put into the array is
# determined by the first position in the shape tuple, in this case 1.
data = np.ndarray(shape=(1, 224, 224, 3), dtype=np.float32)

# Replace this with the path to your image
image = Image.open('test_photo.jpg')

#resize the image to a 224x224 with the same strategy as in TM2:
#resizing the image to be at least 224x224 and then cropping from the center
size = (224, 224)
image = ImageOps.fit(image, size, Image.ANTIALIAS)

#turn the image into a numpy array
image_array = np.asarray(image)

# display the resized image
image.show()

# Normalize the image
normalized_image_array = (image_array.astype(np.float32) / 127.0) - 1

# Load the image into the array
data[0] = normalized_image_array
```

```
# run the inference
prediction = model.predict(data)
print(prediction)
```

3/ Tensorflow Lite:

a/ Android:

```
@Override
protected String getModelPath() {
  return "converted_tflite_quantized/model.tflite";
}

@Override
protected String getLabelPath() {
  return "converted_tflite_quantized/labels.txt";
}
```

b/Coral:

```
from edgetpu.classification.engine import ClassificationEngine
from PIL import Image
import cv2
import re
import os

# the TFLite converted to be used with edgetpu
modelPath = '<PATH_TO_MODEL>'

# The path to labels.txt that was downloaded with your model
labelPath = '<PATH_TO_LABELS>'

# This function parses the labels.txt and puts it in a python dictionary
def loadLabels(labelPath):
    p = re.compile(r'\s*(\d+)(.+)')
    with open(labelPath, 'r', encoding='utf-8') as labelFile:
        lines = (p.match(line).groups() for line in labelFile.readlines())
        return {int(num): text.strip() for num, text in lines}


# This function takes in a PIL Image from any source or path you choose
def classifyImage(image_path, engine):
    # Load and format your image for use with TM2 model
    # image is reformated to a square to match training
    image = Image.open(image_path)
    image.resize((224, 224))

    # Classify and ouptut inference
    classifications = engine.ClassifyWithImage(image)
    return classifications


def main():
    # Load your model onto your Coral Edgetpu
    engine = ClassificationEngine(modelPath)
    labels = loadLabels(labelPath)

    cap = cv2.VideoCapture(0)
    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            break

        # Format the image into a PIL Image so its compatable with Edge TPU
        cv2_im = frame
        pil_im = Image.fromarray(cv2_im)

        # Resize and flip image so its a square and matches training
        pil_im.resize((224, 224))
        pil_im.transpose(Image.FLIP_LEFT_RIGHT)

        # Classify and display image
        results = classifyImage(pil_im, engine)
        cv2.imshow('frame', cv2_im)
        print(results)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    cap.release()
    cv2.destroyAllWindows()

if __name__ == '__main__':
    main()
```