<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>

P5

<div>Teachable Machine Image Model - p5.js and ml5.js</div>

<script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/0.9.0/p5.min.js"></script>

<script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/0.9.0/addons/p5.dom.min.js"></script>

<script src="https://unpkg.com/ml5@0.4.3/dist/ml5.min.js"></script>

<script type="text/javascript">

// Classifier Variable

let classifier;

// Model URL

let imageModelURL = './my\_model/';

// Video

let video;

let flippedVideo;

// To store the classification

let label = "";

// Load the model first

function preload() {

classifier = ml5.imageClassifier(imageModelURL + 'model.json');

}

function setup() {

createCanvas(320, 260);

// Create the video

video = createCapture(VIDEO);

video.size(320, 240);

video.hide();

flippedVideo = ml5.flipImage(video);

// Start classifying

classifyVideo();

}

function draw() {

background(0);

// Draw the video

image(flippedVideo, 0, 0);

// Draw the label

fill(255);

textSize(16);

textAlign(CENTER);

text(label, width / 2, height - 4);

}

// Get a prediction for the current video frame

function classifyVideo() {

flippedVideo = ml5.flipImage(video)

classifier.classify(flippedVideo, gotResult);

flippedVideo.remove();

}

// When we get a result

function gotResult(error, results) {

// If there is an error

if (error) {

console.error(error);

return;

}

// The results are in an array ordered by confidence.

// console.log(results[0]);

label = results[0].label;

// Classifiy again!

classifyVideo();

}

</script>