<클라우드 링크>

<div>Teachable Machine Pose Model</div>

<button type="button" onclick="init()">Start</button>

<div><canvas id="canvas"></canvas></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/pose@0.8/dist/teachablemachine-pose.min.js"></script>

<script type="text/javascript">

// More API functions here:

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

// the link to your model provided by Teachable Machine export panel

const URL = "https://teachablemachine.withgoogle.com/models/jyD8qqg7e/";

let model, webcam, ctx, labelContainer, maxPredictions;

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

// Note: the pose library adds a tmPose object to your window (window.tmPose)

model = await tmPose.load(modelURL, metadataURL);

maxPredictions = model.getTotalClasses();

// Convenience function to setup a webcam

const size = 200;

const flip = true; // whether to flip the webcam

webcam = new tmPose.Webcam(size, size, flip); // width, height, flip

await webcam.setup(); // request access to the webcam

await webcam.play();

window.requestAnimationFrame(loop);

// append/get elements to the DOM

const canvas = document.getElementById("canvas");

canvas.width = size; canvas.height = size;

ctx = canvas.getContext("2d");

labelContainer = document.getElementById("label-container");

for (let i = 0; i < maxPredictions; i++) { // and class labels

labelContainer.appendChild(document.createElement("div"));

}

}

async function loop(timestamp) {

webcam.update(); // update the webcam frame

await predict();

window.requestAnimationFrame(loop);

}

async function predict() {

// Prediction #1: run input through posenet

// estimatePose can take in an image, video or canvas html element

const { pose, posenetOutput } = await model.estimatePose(webcam.canvas);

// Prediction 2: run input through teachable machine classification model

const prediction = await model.predict(posenetOutput);

for (let i = 0; i < maxPredictions; i++) {

const classPrediction =

prediction[i].className + ": " + prediction[i].probability.toFixed(2);

labelContainer.childNodes[i].innerHTML = classPrediction;

}

// finally draw the poses

drawPose(pose);

}

function drawPose(pose) {

if (webcam.canvas) {

ctx.drawImage(webcam.canvas, 0, 0);

// draw the keypoints and skeleton

if (pose) {

const minPartConfidence = 0.5;

tmPose.drawKeypoints(pose.keypoints, minPartConfidence, ctx);

tmPose.drawSkeleton(pose.keypoints, minPartConfidence, ctx);

}

}

}

</script>

<다운로드 링크>

<div>Teachable Machine Pose Model</div>

<button type="button" onclick="init()">Start</button>

<div><canvas id="canvas"></canvas></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/pose@0.8/dist/teachablemachine-pose.min.js"></script>

<script type="text/javascript">

// More API functions here:

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

// the link to your model provided by Teachable Machine export panel

const URL = "./my\_model/";

let model, webcam, ctx, labelContainer, maxPredictions;

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

// Note: the pose library adds a tmPose object to your window (window.tmPose)

model = await tmPose.load(modelURL, metadataURL);

maxPredictions = model.getTotalClasses();

// Convenience function to setup a webcam

const size = 200;

const flip = true; // whether to flip the webcam

webcam = new tmPose.Webcam(size, size, flip); // width, height, flip

await webcam.setup(); // request access to the webcam

await webcam.play();

window.requestAnimationFrame(loop);

// append/get elements to the DOM

const canvas = document.getElementById("canvas");

canvas.width = size; canvas.height = size;

ctx = canvas.getContext("2d");

labelContainer = document.getElementById("label-container");

for (let i = 0; i < maxPredictions; i++) { // and class labels

labelContainer.appendChild(document.createElement("div"));

}

}

async function loop(timestamp) {

webcam.update(); // update the webcam frame

await predict();

window.requestAnimationFrame(loop);

}

async function predict() {

// Prediction #1: run input through posenet

// estimatePose can take in an image, video or canvas html element

const { pose, posenetOutput } = await model.estimatePose(webcam.canvas);

// Prediction 2: run input through teachable machine classification model

const prediction = await model.predict(posenetOutput);

for (let i = 0; i < maxPredictions; i++) {

const classPrediction =

prediction[i].className + ": " + prediction[i].probability.toFixed(2);

labelContainer.childNodes[i].innerHTML = classPrediction;

}

// finally draw the poses

drawPose(pose);

}

function drawPose(pose) {

if (webcam.canvas) {

ctx.drawImage(webcam.canvas, 0, 0);

// draw the keypoints and skeleton

if (pose) {

const minPartConfidence = 0.5;

tmPose.drawKeypoints(pose.keypoints, minPartConfidence, ctx);

tmPose.drawSkeleton(pose.keypoints, minPartConfidence, ctx);

}

}

}

</script>