Artificial Intelligence Module B Unit #4 pre-trained bodyPose

Content

Module B Unit #4 bodyPose with ml5.js

Introduction to pre-trained bodyPose with ml5.js

This is another pre-trained model to use. It picks out the main body points, wrist, knee, elbow, eyes, and so on. You can draw them and watch them move with you; you can also connect them as a skeleton.

One of the uses of this is to analyse movement, especially in sport where you can compare the running action of one runner versus another to identify differences, or in football, golf, or tennis where there may be tweaks that can be made to improve performance.

By default, the `bodyPose` model uses `the MoveNet` version; this has `17` body keypoints, as shown in the diagram below. If you don't specify the version, it defaults to MoveNet. The alternative is `BlazePose,` which has `33` body keypoints and also has a `3D` option (x, y, z). We will use the `MoveNet` for this unit.

Figure 1 BodyPose('moveNet')

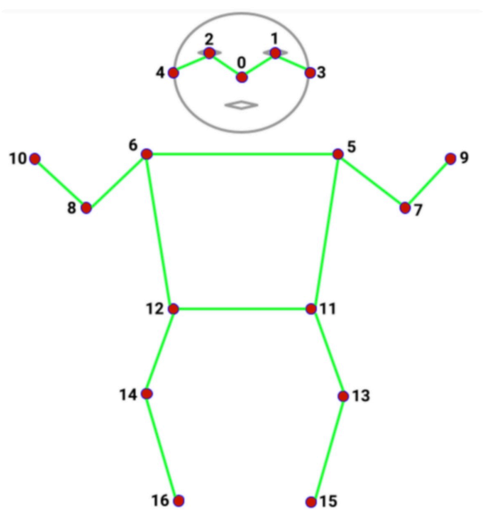
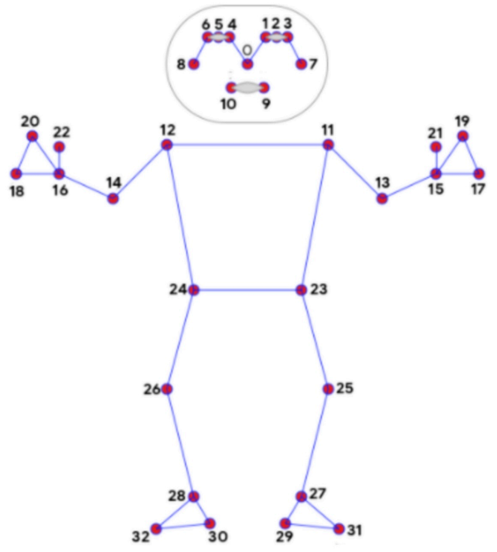


Figure 2 BodyPose('blazePose')



The index.html file

Adding the line of code for ml5.js

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/
1.11.1/p5.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/p5.js/
1.11.1/addons/p5.sound.min.js"></script>
    <script src="https://unpkg.com/ml5@1/dist/ml5.min.js"></
script>
    <link rel="stylesheet" type="text/css" href="style.css">
    <meta charset="utf-8" />


  </head>
  <body>
    <main>
    </main>
    <script src="sketch.js"></script>
  </body>
</html>
```

Sketch B4.1 let us begin

You should be pretty familiar with this now. We start with our webcam video stream, as we have done in the previous units.

```
let video


function setup()
{
  createCanvas(640, 480)
  video = createCapture(VIDEO, {flipped: true})
  video.size(640, 480)
  video.hide()
}


function draw()
{
  background(220)
  image(video, 0, 0, width, height)
}
```

Notes

You should have an image on your screen where the canvas should be. The canvas will be `640` x `480`. You can adjust the separating line between the code and the canvas by dragging it.

Sketch B4.2 preload

We want to preload the pre-trained model so that it is running before we attempt to do anything with it; otherwise, if there is a slight lag, you will get error messages because it cannot find it. We flipped the bodyPose model as well as the video.

```
let video
let bodyPose

async function setup()
{
  ml5.setBackend("webgl")
  bodyPose = await ml5.bodyPose({flipped: true})
  createCanvas(640, 480)
  video = createCapture(VIDEO, {flipped: true})
  video.size(640, 480)
  video.hide()
}

function draw()
{
  background(220)
  image(video, 0, 0, width, height)
}
```

Notes

Nothing new, just missing a few steps we have covered already. Still won't see anything (except yourself) yet.

Sketch B4.3 get the results

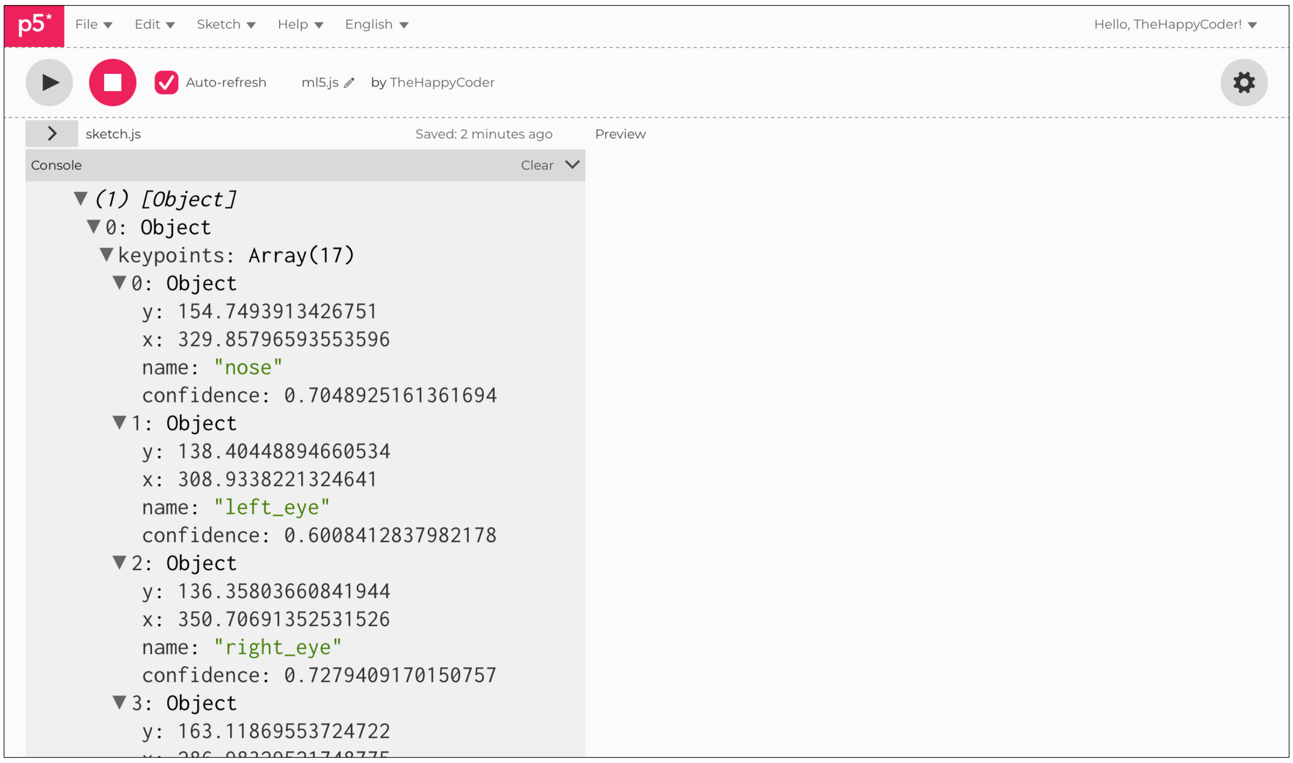
We want the results of the bodyPose. We will console log the results into the poses[] array.

```
let video
let bodyPose
let poses = []

async function setup()
{
  ml5.setBackend("webgl")
  bodyPose = await ml5.bodyPose({flipped: true})
  createCanvas(640, 480)
  video = createCapture(VIDEO, {flipped: true})
  video.size(640, 480)
  video.hide()
  bodyPose.detectStart(video, gotPoses)
}

function draw()
{
  background(220)
  image(video, 0, 0, width, height)
}

function gotPoses(results)
{
  poses = results
  console.log(poses)
}
```

🗒️ Notes

You can see the results in the console (just stop the code running and open up the arrays). They are called `keypoints`, which is important. They contain an (`x, y`) co-ordinates, a description (`name:)`, and a `confidence` score. We can use these values to draw them on the canvas.

Figure B4.3



```
▼(1) [Object]
 ▼0: Object
  ▼keypoints: Array(17)
   ▼0: Object
     y: 154.7493913426751
     x: 329.85796593553596
     name: "nose"
     confidence: 0.7048925161361694
   ▼1: Object
     y: 138.40448894660534
     x: 308.9338221324641
     name: "left_eye"
     confidence: 0.6008412837982178
   ▼2: Object
     y: 136.35803660841944
     x: 350.70691352531526
     name: "right_eye"
     confidence: 0.7279409170150757
   ▼3: Object
     y: 163.11869553724722
```

Sketch B4.4 keypoints

Let's draw these keypoints on the canvas. We have all the results in the poses[] array. We are going to loop through them and pull out the co-ordinates of all the keypoints.

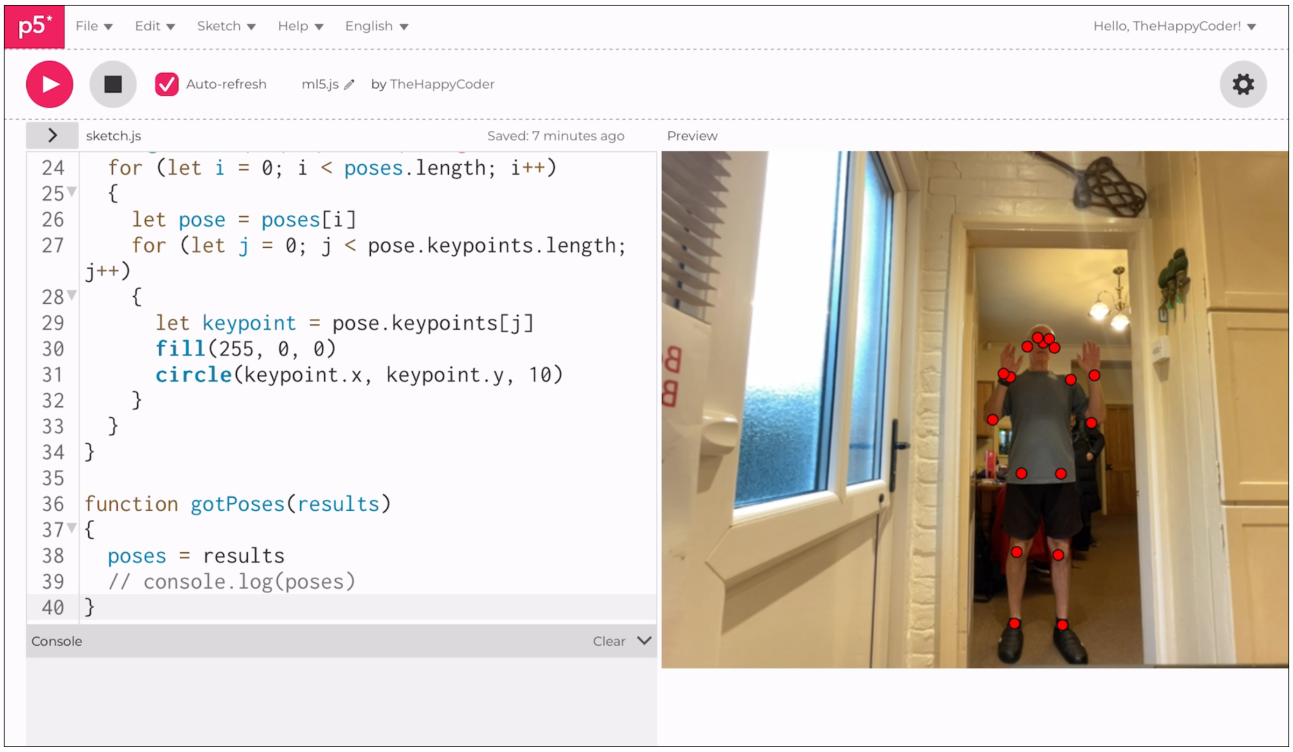
❗ Remove the console log

```
let video
let bodyPose
let poses = []

async function setup()
{
  ml5.setBackend("webgl")
  bodyPose = await ml5.bodyPose({flipped: true})
  createCanvas(640, 480)
  video = createCapture(VIDEO, {flipped: true})
  video.size(640, 480)
  video.hide()
  bodyPose.detectStart(video, gotPoses)
}

function draw()
{
  background(220)
  image(video, 0, 0, width, height)
  for (let i = 0; i < poses.length; i++)
  {
    let pose = poses[i]
    for (let j = 0; j < pose.keypoints.length; j++)
    {
      let keypoint = pose.keypoints[j]
      fill(255, 0, 0)
      circle(keypoint.x, keypoint.y, 10)
    }
  }
}

function gotPoses(results)
{
  poses = results
  // console.log(poses)
}
```

🗒️ Notes

In the console log, you can see the number of `keypoints`.

Figure B4.4



```
  for (let i = 0; i < poses.length; i++)
  {
    let pose = poses[i]
    for (let j = 0; j < pose.keypoints.length; j++)
    {
      let keypoint = pose.keypoints[j]
      fill(255, 0, 0)
      circle(keypoint.x, keypoint.y, 10)
    }
  }
}

function gotPoses(results)
{
  poses = results
  // console.log(poses)
}
```

Sketch B4.5 confidence

We draw all the points, whether you can see the parts of the body or not. We can add a conditional statement to only draw the point if the confidence score is above `0.1`.

```
let video
let bodyPose
let poses = []

async function setup()
{
  ml5.setBackend("webgl")
  bodyPose = await ml5.bodyPose({flipped: true})
  createCanvas(640, 480)
  video = createCapture(VIDEO, {flipped: true})
  video.size(640, 480)
  video.hide()
  bodyPose.detectStart(video, gotPoses)
}

function draw()
{
  background(220)
  image(video, 0, 0, width, height)
  for (let i = 0; i < poses.length; i++)
  {
    let pose = poses[i]
    for (let j = 0; j < pose.keypoints.length; j++)
    {
      let keypoint = pose.keypoints[j]
      if (keypoint.confidence > 0.1)
      {
        fill(255, 0, 0)
        circle(keypoint.x, keypoint.y, 10)
      }
    }
  }
}

function gotPoses(results)
{
  poses = results
  // console.log(poses)
}
```

🗒️ Notes

Now you should only get ones that are relevant to what you can see.

🌻 Challenge

Change the confidence score.

🛠️ Code Explanation

`if (keypoint.confidence > 0.1)`	Another attribute found in the array, here we check to see if the key point confidence is above 0.1 for each keypoint

Sketch B4.6 skeleton

Let's draw the skeleton by drawing a line between points. Notice that we aren't just joining all the points, only the relevant ones, using the `getSkeleton()` function as part of the `bodyPose` model. There is a lot of code to get your head around and would take a lot of explaining. Just understand that every point has a corresponding point, and the inner nested loop (containing the `j` variable) goes through all of them. I suggest console logging to see what is happening.

```
let video
let bodyPose
let poses = []
let connections

async function setup()
{
  ml5.setBackend("webgl")
  bodyPose = await ml5.bodyPose({flipped: true})
  createCanvas(640, 480)
  video = createCapture(VIDEO, {flipped: true})
  video.size(640, 480)
  video.hide()
  bodyPose.detectStart(video, gotPoses)
  connections = bodyPose.getSkeleton()
}

function draw()
{
  background(220)
  image(video, 0, 0, width, height)
  for (let i = 0; i < poses.length; i++)
  {
    let pose = poses[i]
    for (let j = 0; j < pose.keypoints.length; j++)
    {
      let keypoint = pose.keypoints[j]
      // if (keypoint.confidence > 0.1)
      // {
      //   fill(255, 0, 0)
      //   circle(keypoint.x, keypoint.y, 10)
      // }
    }
  }
  for (let i = 0; i < poses.length; i++)
  {
    let pose = poses[i]
    for (let j = 0; j < connections.length; j++)
    {
      let pointAIndex = connections[j][0]
      let pointBIndex = connections[j][1]
      let pointA = pose.keypoints[pointAIndex]
      let pointB = pose.keypoints[pointBIndex]
      if (pointA.confidence > 0.1 && pointB.confidence > 0.1)
      {
        stroke(255)
        strokeWeight(3)
        line(pointA.x, pointA.y, pointB.x, pointB.y)
      }
    }
  }
}

function gotPoses(results)
{
  poses = results
}
```

🗒️ Notes

You should have the white lines joining most of the red circles. There are a lot of longer-named variables, which makes it difficult to read, but take your time to try to understand; otherwise, just know how to use it.

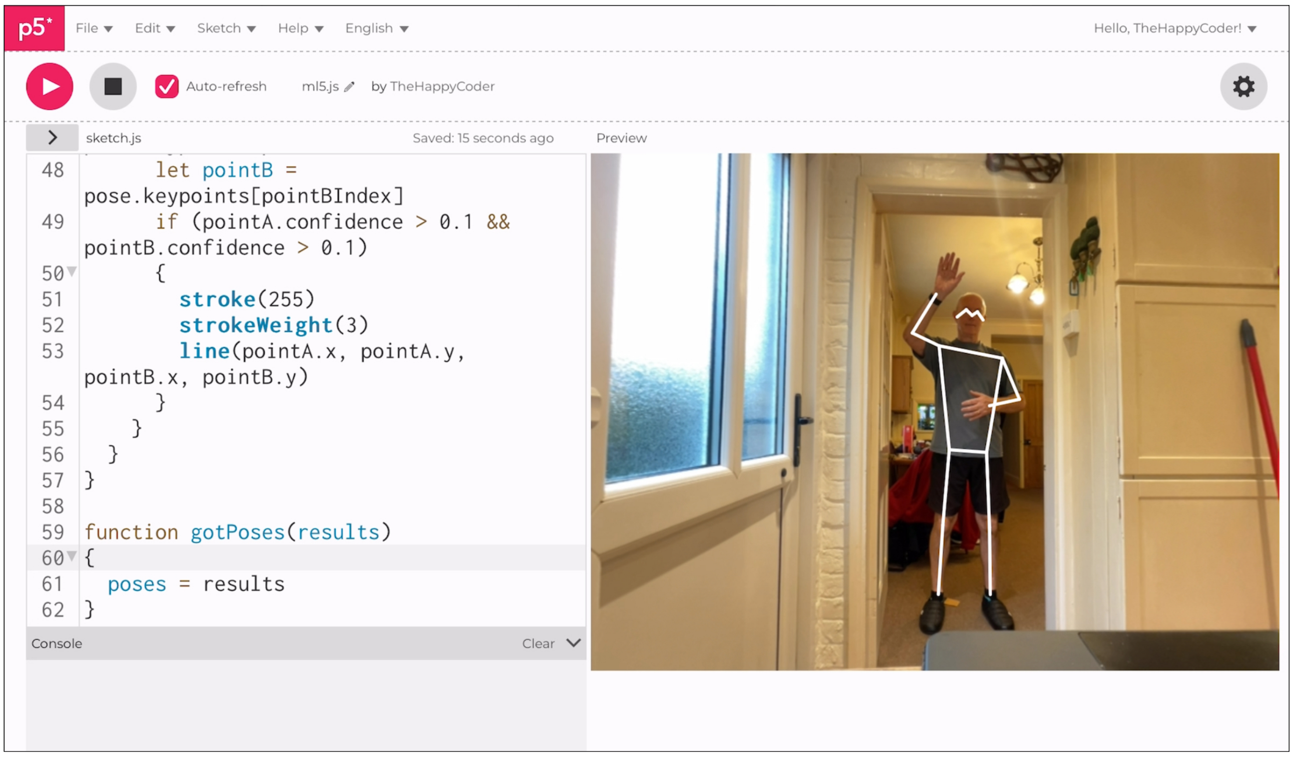
🌻 Challenge

As I said earlier, to really start to understand the logic behind the nested loop, I recommend using the console log to explore.

🛠️ Code Explanation

`connections = bodyPose.getSkeleton()`	The connections hold the skeleton connection points, this is so that they are joined together in a logical way rather than all of them joining each other.

Figure B4.6



```
      let pointB =
pose.keypoints[pointBIndex]
      if (pointA.confidence > 0.1 &&
pointB.confidence > 0.1)
      {
        stroke(255)
        strokeWeight(3)
        line(pointA.x, pointA.y,
pointB.x, pointB.y)
      }
    }
  }
}

function gotPoses(results)
{
  poses = results
}
```