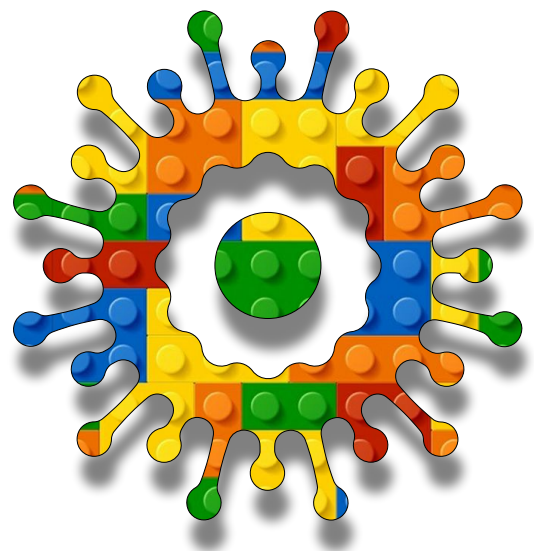


Algorithmic
Intelligence
Module B
Unit #5
pretrained
speech





Module B Unit #5 speech classification with ml5.js

Introduction to speech classification

The index.html file

Sketch B5.1	the preload
Sketch B5.2	classifying
Sketch B5.3	got the results
Sketch B5.4	display
Sketch B5.5	a moving ball
Sketch B5.6	a splash of colour and text



Introduction to pretrained soundClassifier with ml5.js

In this unit, we will demonstrate pre-trained speech classification using the **SpeechCommands18w** pretrained model. We will look at the basic model and see how good it is, and then use some of those commands to move an object around. There are **18** words, and they are as follows:

Zero, One, Two, Three, Four, Five, Six, Seven, Eight, Nine, Up, Down, Left, Right, Go, Stop, Yes, No

You will need to allow access to the **microphone** on your computer (assuming there is one). The results can be a bit sketchy, but it is a good demonstration of what can be achieved. One real-world application is a wake word similar to a smart speaker.



The index.html file

Adding the code line 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 B5.1 the preload starting sketch

We will preload the pretrained model. We want it to have a confidence score of **0.7** or higher. We load **SpeechCommands18w** from ml5.js.

```
let classifier

async function setup()
{
  ml5.setBackend("webgl")
  let options = {probabilityThreshold: 0.7}
  classifier = await ml5.soundClassifier("SpeechCommands18w")
  createCanvas(400, 400)
}

function draw()
{
  background(220)
}
```



Notes

Nothing is meant to happen yet; all we have done is download the model.



Code Explanation

<code>let options = {probabilityThreshold: 0.7}</code>	The probabilityThreshold is a built in function
<code>classifier = await ml5.soundClassifier("SpeechCommands18w")</code>	This is a speech classifier that uses the SpeechCommands18W pretrained words



Sketch B5.2 classifying

We want to start the pretrained model and classify the sounds/words using the `classifyStart()` function with a callback.

```
let classifier

async function setup()
{
  ml5.setBackend("webgl")
  let options = {probabilityThreshold: 0.7}
  classifier = await ml5.soundClassifier("SpeechCommands18w")
  createCanvas(400, 400)
  classifier.classifyStart(gotResults)
}

function draw()
{
  background(220)
}
```



Notes

! You will get an error message if you run it now.
We will want a `gotResults()` callback function.



Code Explanation

```
classifier.classifyStart(gotResults)
```

Starts the classification of the sounds



Sketch B5.3 the got the results

Once we have the results, we need the callback function `gotResults()` and predict the label which will appear as the top, or highest, confidence score at `index[0]`. We will add the console log temporarily to see the results.

```
let classifier
let predictedWord = ""

async function setup()
{
  ml5.setBackend("webgl")
  let options = {probabilityThreshold: 0.7}
  classifier = await ml5.soundClassifier("SpeechCommands18w")
  createCanvas(400, 400)
  classifier.classifyStart(gotResults)
}

function draw()
{
  background(220)
}

function gotResults(results)
{
  predictedWord = results[0].label
  console.log(predictedWord)
}
```



Notes

It will pick up every sound and try to classify it. The output also depends on what data it is trained on, whose voices, and what accents (American, British, or a mixture). Another factor may be the sensitivity of your built-in microphone.



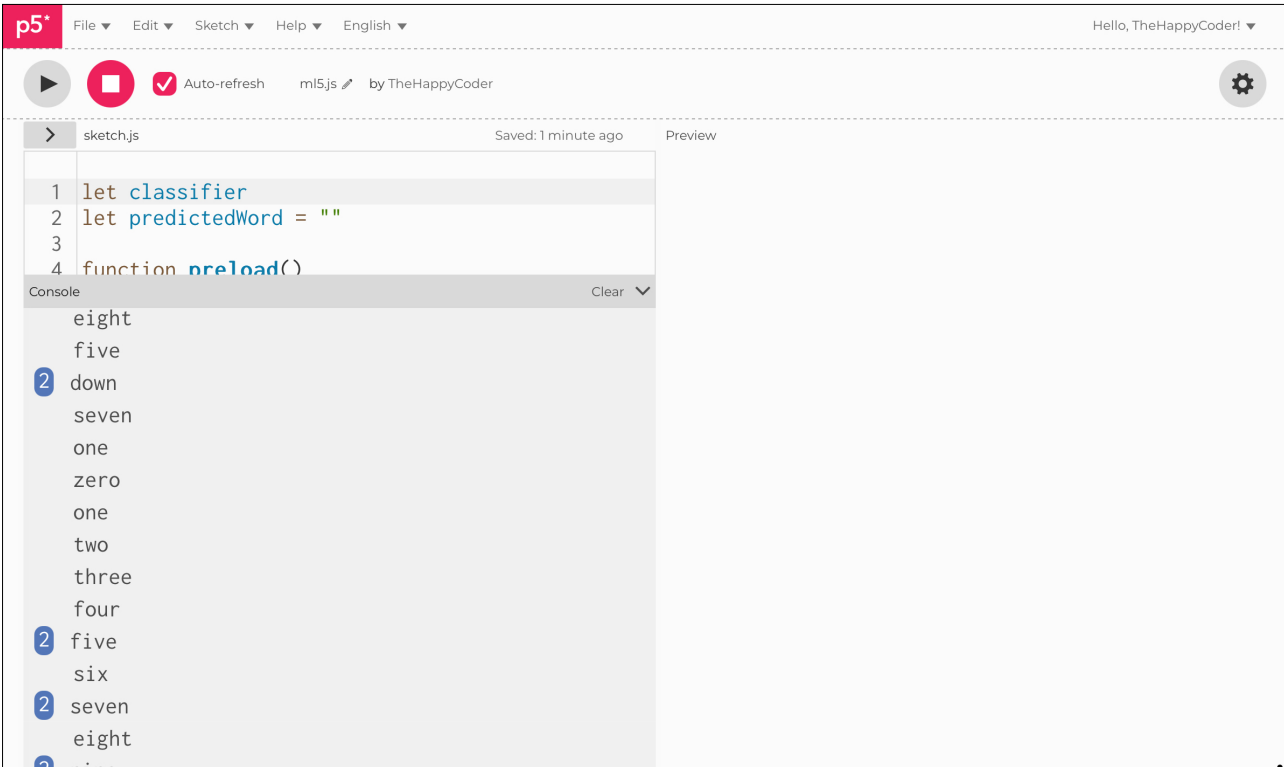
Challenge

Try different accents or shouting (quietly).

Code Explanation

<code>let predictedWord = ""</code>	Declare and initialise an empty string variable
<code>predictedWord = results[0].label</code>	Finds the first label with the highest probability

Figure B5.3



The screenshot shows the p5.js IDE interface. The code editor displays the following code:

```
1 let classifier
2 let predictedWord = ""
3
4 function preload()
```

The console output shows a list of words: eight, five, down, seven, one, zero, one, two, three, four, five, six, seven, eight. The word 'down' is highlighted with a blue circle and the number '2' next to it, indicating it is the predicted word.



Sketch B5.5 a moving ball

Removing the text results in `draw()`. We will move the ball around the canvas with our voice. Add the `edges()` function so that if it goes off the edge, you don't lose it entirely. We will use the commands **UP**, **DOWN**, **LEFT**, **RIGHT**, and **STOP**.

```
let classifier
let predictedWord = ""

let x = 200
let y = 200

async function setup()
{
  ml5.setBackend("webgl")
  let options = {probabilityThreshold: 0.7}
  classifier = await ml5.soundClassifier("SpeechCommands18w")
  createCanvas(400, 400)
  classifier.classifyStart(gotResults)
}

function draw()
{
  background(220)

  if (predictedWord == "stop")
  {
    circle(x, y, 50)
    x = x
    y = y
  }
  else if (predictedWord == "up")
  {
    circle(x, y, 50)
    y -= 5
  }
  else if (predictedWord == "down")
  {
    circle(x, y, 50)
    y += 5
  }
}
```

```
}  
else if (predictedWord == "left")  
{  
  circle(x, y, 50)  
  x -= 5  
}  
else if (predictedWord == "right")  
{  
  circle(x, y, 50)  
  x += 5  
}  
else  
{  
  circle(x, y, 50)  
}  
edges()  
}
```

```
function gotResults(results)  
{  
  predictedWord = results[0].label  
  console.log(predictedWord)  
}
```

```
function edges()  
{  
  if (x > width)  
  {  
    x = 0  
  }  
  if (x < 0)  
  {  
    x = width  
  }  
  if (y > height)  
  {  
    y = 0  
  }  
}
```

```
if (y < 0)
{
  y = height
}
}
```

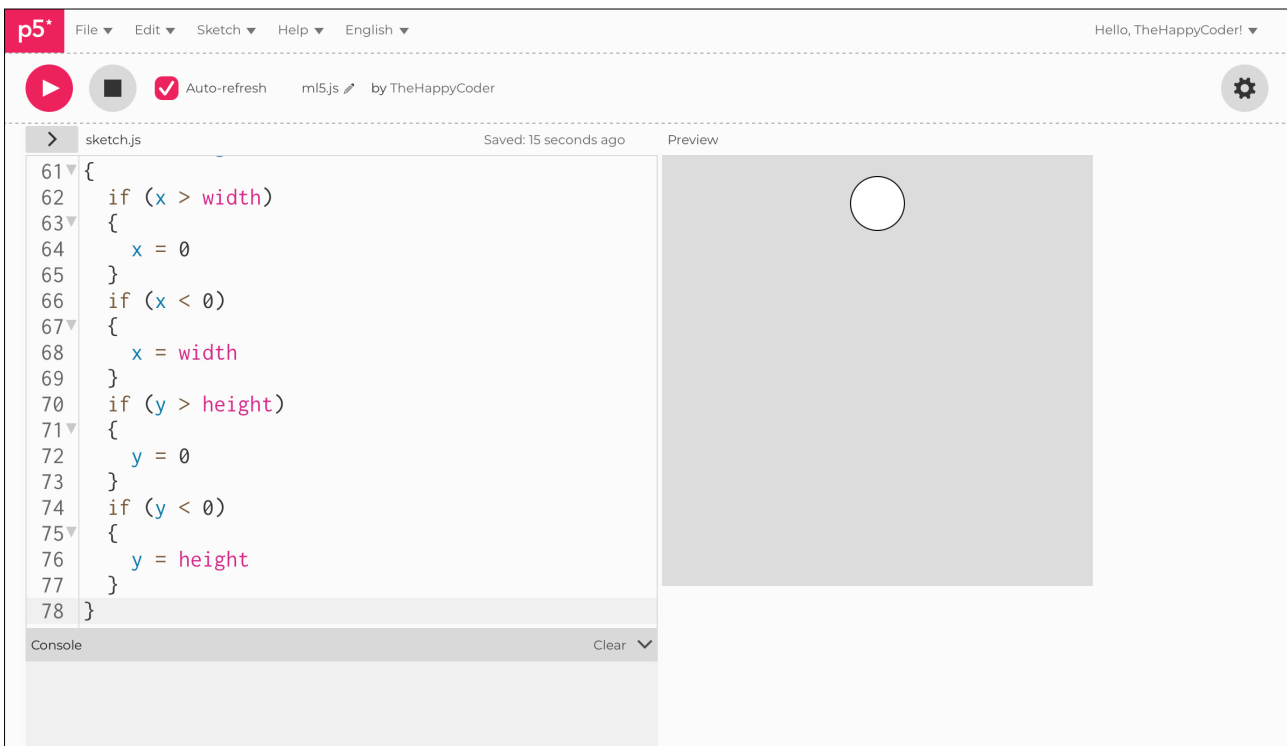
Notes

You should be able to move the circle around with your voice. There are quite a few lines of code, but it does not require much typing. The `edge()` function is standard in most games so that the object stays somewhere on the canvas. It is pretty self-evident.

Challenge

You could change the `edges()` function so that it stops at the edge rather than reappearing on the opposite edge.

Figure B5.5





Sketch B5.6 a splash of colour and text

Adding a splash of colour and the text returns on the canvas.

```
let classifier
let predictedWord = ""
let x = 200
let y = 200

async function setup()
{
  ml5.setBackend("webgl")
  let options = {probabilityThreshold: 0.7}
  classifier = await ml5.soundClassifier("SpeechCommands18w")
  createCanvas(400, 400)
  classifier.classifyStart(gotResults)
}

function draw()
{
  background(220, 0, 200)
  if (predictedWord !== "")
  {
    fill(0)
    textAlign(CENTER, CENTER)
    textSize(64)
    text(predictedWord, width/2, height/2)
  }
  fill(200, 200, 10)
  noStroke()
  if (predictedWord == "stop")
  {
    circle(x, y, 50)
    x = x
    y = y
  }
  else if (predictedWord == "up")
  {
```

```
    circle(x, y, 50)
    y -= 5
  }
  else if (predictedWord == "down")
  {
    circle(x, y, 50)
    y += 5
  }
  else if (predictedWord == "left")
  {
    circle(x, y, 50)
    x -= 5
  }
  else if (predictedWord == "right")
  {
    circle(x, y, 50)
    x += 5
  }
  else
  {
    circle(x, y, 50)
  }
  edges()
}

function gotResults(results)
{
  predictedWord = results[0].label
  console.log(predictedWord)
}

function edges()
{
  if (x > width)
  {
    x = 0
  }
  if (x < 0)
```

```
{
  x = width
}
if (y > height)
{
  y = 0
}
if (y < 0)
{
  y = height
}
}
```



Notes

The accuracy will be the same, which is not great in my case; you may have better results. The accuracy will be affected by background noise, the quality of your microphone, and possibly your accent, amongst other things. Have a bit of fun with it.



Challenge

1. Try changing the `probabilityThreshold` value.
2. What could you do with the numbers?
3. Could you filter out the words you don't want?



Code Explanation

```
if (predictedWord !== "")
```

If there is NOT an empty string, in other words there IS a word it recognises

Figure B5.6

