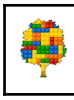
Module 7
Colour Predictor
ml5.js

Content

Module 7: Colour Predictor with ml5.js

Introduction to module 7 Colour Predictor

Another seemingly simple exercise to train the model to identify reddish , greenish and blueish colours. For this we will provide a small dataset in the form of a json style file.



Sketch 7.1 let's create the data

We are going to create three labels 'red-ish', 'green-ish', and 'blue-ish', for each of these three labels we are going to give them the RGB numbers that are pretty close, eg red is (255, 0, 0), (254, 0, 0) and (253, 0, 0) and so on for green and blue

```
let data = [
  { r: 255, g: 0, b: 0, colour: "red-ish" },
  { r: 254, g: 0, b: 0, colour: "red-ish" },
  { r: 253, g: 0, b: 0, colour: "red-ish" },
  { r: 0, g: 255, b: 0, colour: "green-ish" },
  { r: 0, g: 254, b: 0, colour: "green-ish" },
  { r: 0, g: 253, b: 0, colour: "green-ish" },
  { r: 0, g: 0, b: 255, colour: "blue-ish" },
  { r: 0, g: 0, b: 254, colour: "blue-ish" },
  { r: 0, g: 0, b: 253, colour: "blue-ish" },
]

function setup()
{
  createCanvas(400, 400)
}

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

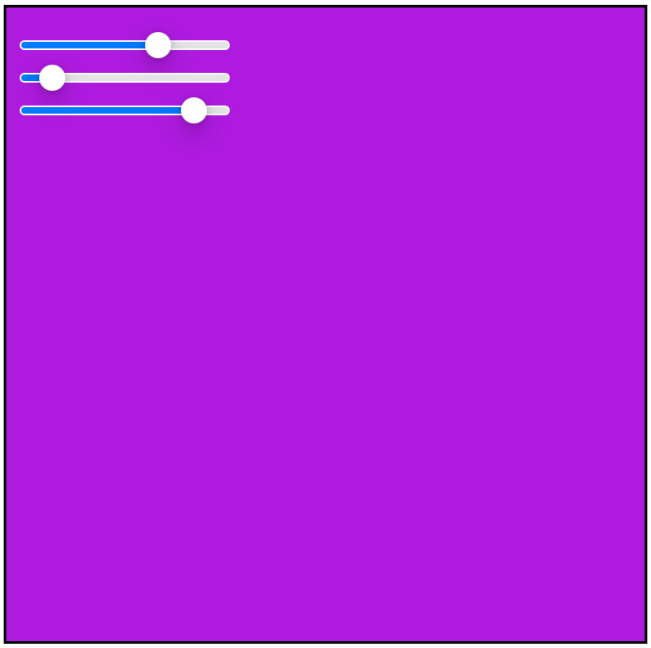
Sketch 7.2 Sliders

We are going to create three sliders to control the colour of the background. We will give the background an initial colour red.

```
let data = [
  { r: 255, g: 0, b: 0, colour: "red-ish" },
  { r: 254, g: 0, b: 0, colour: "red-ish" },
  { r: 253, g: 0, b: 0, colour: "red-ish" },
  { r: 0, g: 255, b: 0, colour: "green-ish" },
  { r: 0, g: 254, b: 0, colour: "green-ish" },
  { r: 0, g: 253, b: 0, colour: "green-ish" },
  { r: 0, g: 0, b: 255, colour: "blue-ish" },
  { r: 0, g: 0, b: 254, colour: "blue-ish" },
  { r: 0, g: 0, b: 253, colour: "blue-ish" },
]

let r
let g
let b
let rSlider
let gSlider
let bSlider

function setup()
{
  createCanvas(400, 400)
  rSlider = createSlider(0, 255, 255).position(10, 20)
  gSlider = createSlider(0, 255, 0).position(10, 40)
  bSlider = createSlider(0, 255, 0).position(10, 60)
}

function draw()
```

```
{
  r = rSlider.value()
  g = gSlider.value()
  b = bSlider.value()
  background(r, g, b)
}
```

Three slider, red, green and blue, move the sliders to change the background colour



Sketch 7.3 adding the neural network

Let's introduce our neural network (ml5.js). Don't forget to add ml5.js to the index.html file.

```
let data = [
  { r: 255, g: 0, b: 0, colour: "red-ish" },
  { r: 254, g: 0, b: 0, colour: "red-ish" },
  { r: 253, g: 0, b: 0, colour: "red-ish" },
  { r: 0, g: 255, b: 0, colour: "green-ish" },
  { r: 0, g: 254, b: 0, colour: "green-ish" },
  { r: 0, g: 253, b: 0, colour: "green-ish" },
  { r: 0, g: 0, b: 255, colour: "blue-ish" },
  { r: 0, g: 0, b: 254, colour: "blue-ish" },
  { r: 0, g: 0, b: 253, colour: "blue-ish" },
]

let r
let g
let b
let rSlider
let gSlider
let bSlider
let nn

function setup()
{
  createCanvas(400, 400)
  rSlider = createSlider(0, 255, 255).position(10, 20)
  gSlider = createSlider(0, 255, 0).position(10, 40)
  bSlider = createSlider(0, 255, 0).position(10, 60)
  let options = {
    task: 'classification',
```

```
    debug: true
  }
  nn = ml5.neuralNetwork(options)
}

function draw()
{
  r = rSlider.value()
  g = gSlider.value()
  b = bSlider.value()
  background(r, g, b)
}
```

Sketch 7.4 adding the data

Now to add the data to the neural network

```
let data = [
  { r: 255, g: 0, b: 0, colour: "red-ish" },
  { r: 254, g: 0, b: 0, colour: "red-ish" },
  { r: 253, g: 0, b: 0, colour: "red-ish" },
  { r: 0, g: 255, b: 0, colour: "green-ish" },
  { r: 0, g: 254, b: 0, colour: "green-ish" },
  { r: 0, g: 253, b: 0, colour: "green-ish" },
  { r: 0, g: 0, b: 255, colour: "blue-ish" },
  { r: 0, g: 0, b: 254, colour: "blue-ish" },
  { r: 0, g: 0, b: 253, colour: "blue-ish" },
]

let r
let g
let b
let rSlider
let gSlider
let bSlider
let nn

function setup()
{
  createCanvas(400, 400)
  rSlider = createSlider(0, 255, 255).position(10, 20)
  gSlider = createSlider(0, 255, 0).position(10, 40)
  bSlider = createSlider(0, 255, 0).position(10, 60)
  let options = {
    task: 'classification',
    debug: true
```

```
  }
  nn = ml5.neuralNetwork(options)
  for (let i = 0; i < data.length; i++)
  {
    let item = data[i]
    let inputs = [item.r, item.g, item.b]
    let outputs = [item.colour]
    nn.addData(inputs, outputs)
  }
}

function draw()
{
  r = rSlider.value()
  g = gSlider.value()
  b = bSlider.value()
  background(r, g, b)
}
```

Sketch 7.5 normalising the data

Normalising the data because we don't want large and time numbers in the neural network calculations. We normalise the between 0 and 1 (or -1 and +1)

```
let data = [
  { r: 255, g: 0, b: 0, colour: "red-ish" },
  { r: 254, g: 0, b: 0, colour: "red-ish" },
  { r: 253, g: 0, b: 0, colour: "red-ish" },
  { r: 0, g: 255, b: 0, colour: "green-ish" },
  { r: 0, g: 254, b: 0, colour: "green-ish" },
  { r: 0, g: 253, b: 0, colour: "green-ish" },
  { r: 0, g: 0, b: 255, colour: "blue-ish" },
  { r: 0, g: 0, b: 254, colour: "blue-ish" },
  { r: 0, g: 0, b: 253, colour: "blue-ish" },
]

let r
let g
let b
let rSlider
let gSlider
let bSlider
let nn

function setup()
{
  createCanvas(400, 400)
  rSlider = createSlider(0, 255, 255).position(10, 20)
  gSlider = createSlider(0, 255, 0).position(10, 40)
  bSlider = createSlider(0, 255, 0).position(10, 60)
  let options = {
```

```
    task: 'classification',
    debug: true
  }
  nn = ml5.neuralNetwork(options)
  for (let i = 0; i < data.length; i++)
  {
    let item = data[i]
    let inputs = [item.r, item.g, item.b]
    let outputs = [item.colour]
    nn.addData(inputs, outputs)
  }
  nn.normalizeData()
}

function draw()
{
  r = rSlider.value()
  g = gSlider.value()
  b = bSlider.value()
  background(r, g, b)
}
```

Sketch 7.6 training the neural network

We will now train our neural network, we will set the batch size to 16 and the epochs to 64 for starters. We will leave the finishedTraining() function empty for now (if you prefer just add console.log('finished training'))

```
let data = [
  { r: 255, g: 0, b: 0, colour: "red-ish" },
  { r: 254, g: 0, b: 0, colour: "red-ish" },
  { r: 253, g: 0, b: 0, colour: "red-ish" },
  { r: 0, g: 255, b: 0, colour: "green-ish" },
  { r: 0, g: 254, b: 0, colour: "green-ish" },
  { r: 0, g: 253, b: 0, colour: "green-ish" },
  { r: 0, g: 0, b: 255, colour: "blue-ish" },
  { r: 0, g: 0, b: 254, colour: "blue-ish" },
  { r: 0, g: 0, b: 253, colour: "blue-ish" },
]

let r
let g
let b
let rSlider
let gSlider
let bSlider
let nn

function setup()
{
  createCanvas(400, 400)
  rSlider = createSlider(0, 255, 255).position(10, 20)
  gSlider = createSlider(0, 255, 0).position(10, 40)
  bSlider = createSlider(0, 255, 0).position(10, 60)
  let options = {
```

```
    task: 'classification',
    debug: true
  }
  nn = ml5.neuralNetwork(options)
  for (let i = 0; i < data.length; i++)
  {
    let item = data[i]
    let inputs = [item.r, item.g, item.b]
    let outputs = [item.colour]
    nn.addData(inputs, outputs)
  }
  nn.normalizeData()
  const trainingOptions = {
    batchSize: 16,
    epochs: 128
  }
  nn.train(trainingOptions, finishedTraining)
}

function finishedTraining()
{

}

function draw()
{
  r = rSlider.value()
  g = gSlider.value()
  b = bSlider.value()
  background(r, g, b)
}
```

Sketch 7.7 classifying the colour

We can now classify the colour as red-ish, green-ish or blue-ish. We have the results sent to the handleResults() function and we can have a look at the results to understand what we want. We console.log the results and in the console you should get the following (from the default)

```
let data = [
  { r: 255, g: 0, b: 0, colour: "red-ish" },
  { r: 254, g: 0, b: 0, colour: "red-ish" },
  { r: 253, g: 0, b: 0, colour: "red-ish" },
  { r: 0, g: 255, b: 0, colour: "green-ish" },
  { r: 0, g: 254, b: 0, colour: "green-ish" },
  { r: 0, g: 253, b: 0, colour: "green-ish" },
  { r: 0, g: 0, b: 255, colour: "blue-ish" },
  { r: 0, g: 0, b: 254, colour: "blue-ish" },
  { r: 0, g: 0, b: 253, colour: "blue-ish" },
]

let r
let g
let b
let rSlider
let gSlider
let bSlider
let nn

function setup()
{
  createCanvas(400, 400)
  rSlider = createSlider(0, 255, 255).position(10, 20)
  gSlider = createSlider(0, 255, 0).position(10, 40)
  bSlider = createSlider(0, 255, 0).position(10, 60)
```

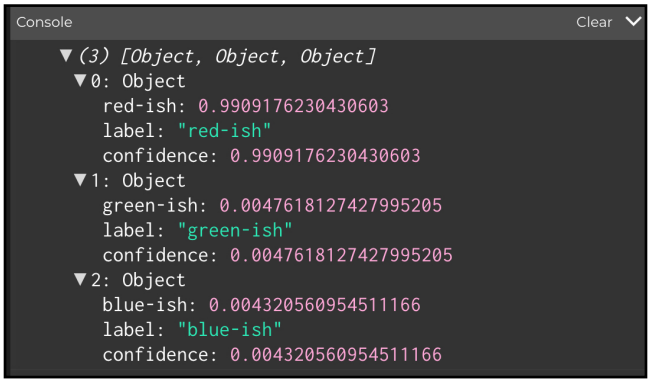
```
  let options = {
    task: 'classification',
    debug: true
  }
  nn = ml5.neuralNetwork(options)
  for (let i = 0; i < data.length; i++)
  {
    let item = data[i]
    let inputs = [item.r, item.g, item.b]
    let outputs = [item.colour]
    nn.addData(inputs, outputs)
  }
  nn.normalizeData()
  const trainingOptions = {
    batchSize: 16,
    epochs: 128
  }
  nn.train(trainingOptions, finishedTraining)
}

function finishedTraining()
{
  classify()
}

function classify()
{
  const input = [r, g, b]
  nn.classify(input, handleResults)
}

function handleResults(results)
{
```

```
  console.log(results)
}


function draw()
{
  r = rSlider.value()
  g = gSlider.value()
  b = bSlider.value()
  background(r, g, b)
}
```

Notes

You get an array with three objects
Each object has a label and confidence score
The highest confidence score is the top one result[0]
The next highest is result[1]
The third is result[2]
We want the top one

The console.log results, click on the little arrows to expand



Sketch 7.8 the best prediction

Extracting the top result with the highest confidence and then printing the result as text on the canvas. Remove the console.log(). We initialise the label as a string (let label = " "). We add the classify() function in the handleResults() function as check loop, so that every time you move the sliders you get an instant update.

```
let data = [
  { r: 255, g: 0, b: 0, colour: "red-ish" },
  { r: 254, g: 0, b: 0, colour: "red-ish" },
  { r: 253, g: 0, b: 0, colour: "red-ish" },
  { r: 0, g: 255, b: 0, colour: "green-ish" },
  { r: 0, g: 254, b: 0, colour: "green-ish" },
  { r: 0, g: 253, b: 0, colour: "green-ish" },
  { r: 0, g: 0, b: 255, colour: "blue-ish" },
  { r: 0, g: 0, b: 254, colour: "blue-ish" },
  { r: 0, g: 0, b: 253, colour: "blue-ish" },
]

let r
let g
let b
let rSlider
let gSlider
let bSlider
let nn
let label = " "

function setup()
{
  createCanvas(400, 400)
  rSlider = createSlider(0, 255, 255).position(10, 20)
  gSlider = createSlider(0, 255, 0).position(10, 40)
```

```
  bSlider = createSlider(0, 255, 0).position(10, 60)
  let options = {
    task: 'classification',
    debug: true
  }
  nn = ml5.neuralNetwork(options)
  for (let i = 0; i < data.length; i++)
  {
    let item = data[i]
    let inputs = [item.r, item.g, item.b]
    let outputs = [item.colour]
    nn.addData(inputs, outputs)
  }
  nn.normalizeData()
  const trainingOptions = {
    batchSize: 16,
    epochs: 128
  }
  nn.train(trainingOptions, finishedTraining)
}

function finishedTraining()
{
  classify()
}

function classify()
{
  const input = [r, g, b]
  nn.classify(input, handleResults)
}

function handleResults(results)
```

```
{
  label = results[0].label
  classify()
}

function draw()
{
  r = rSlider.value()
  g = gSlider.value()
  b = bSlider.value()
  background(r, g, b)
  textSize(64)
  text(label, width/4, height/2)
}
```

Notes

You will perhaps notice that the results aren't perfect despite the loss function looking so good. We have only provided a small amount of data but also we haven't offered any change to the hyperparameters which is something you could do.

Move the sliders to generate the result