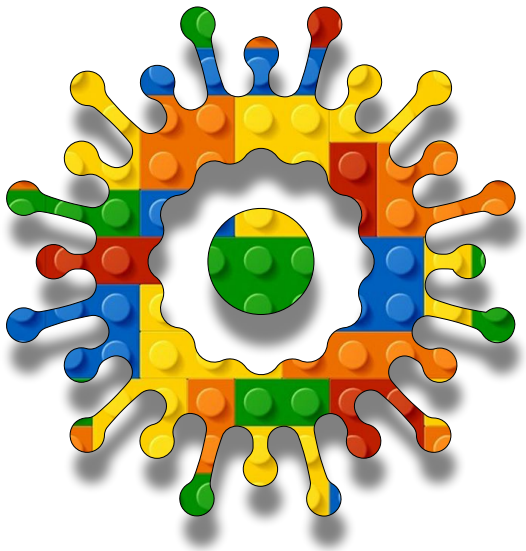
Algorithmic Intelligence Module C Unit #4 transformer image detection



Content

Module C Unit #4 transformer image detection

This looks for objects it can detect and puts a bounding box (plus label) around that object. This is useful if you want to train a model to detect specific objects.

Upload the dog image as we have done before, use your own or click on the button for one provided. We just call it dog (remember to rename it in the file).

Figure 1: download dog image

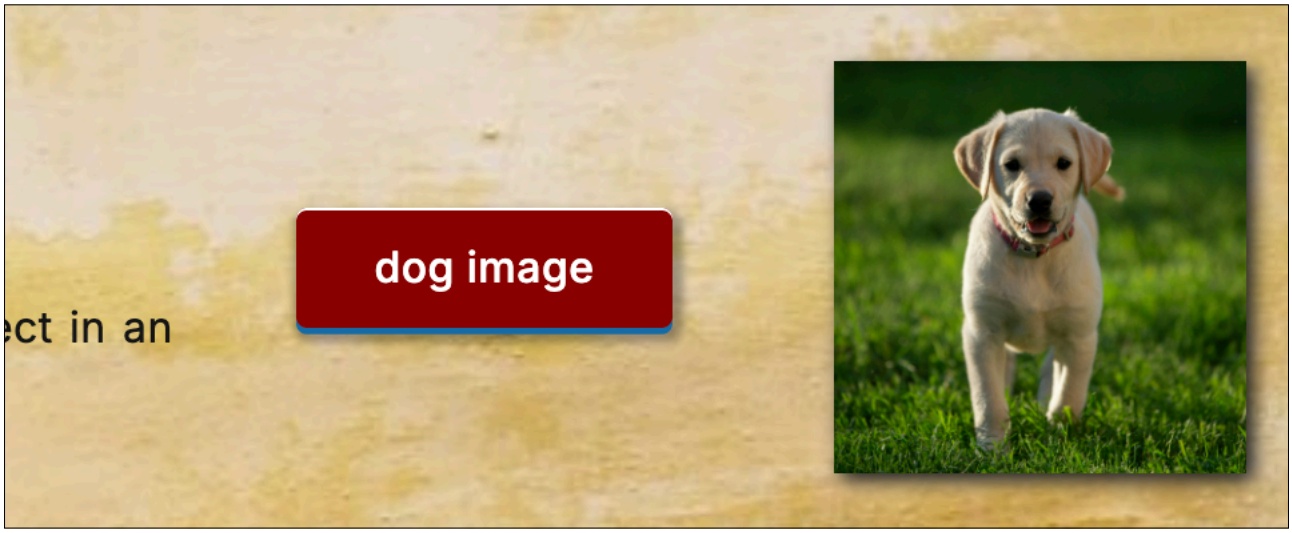
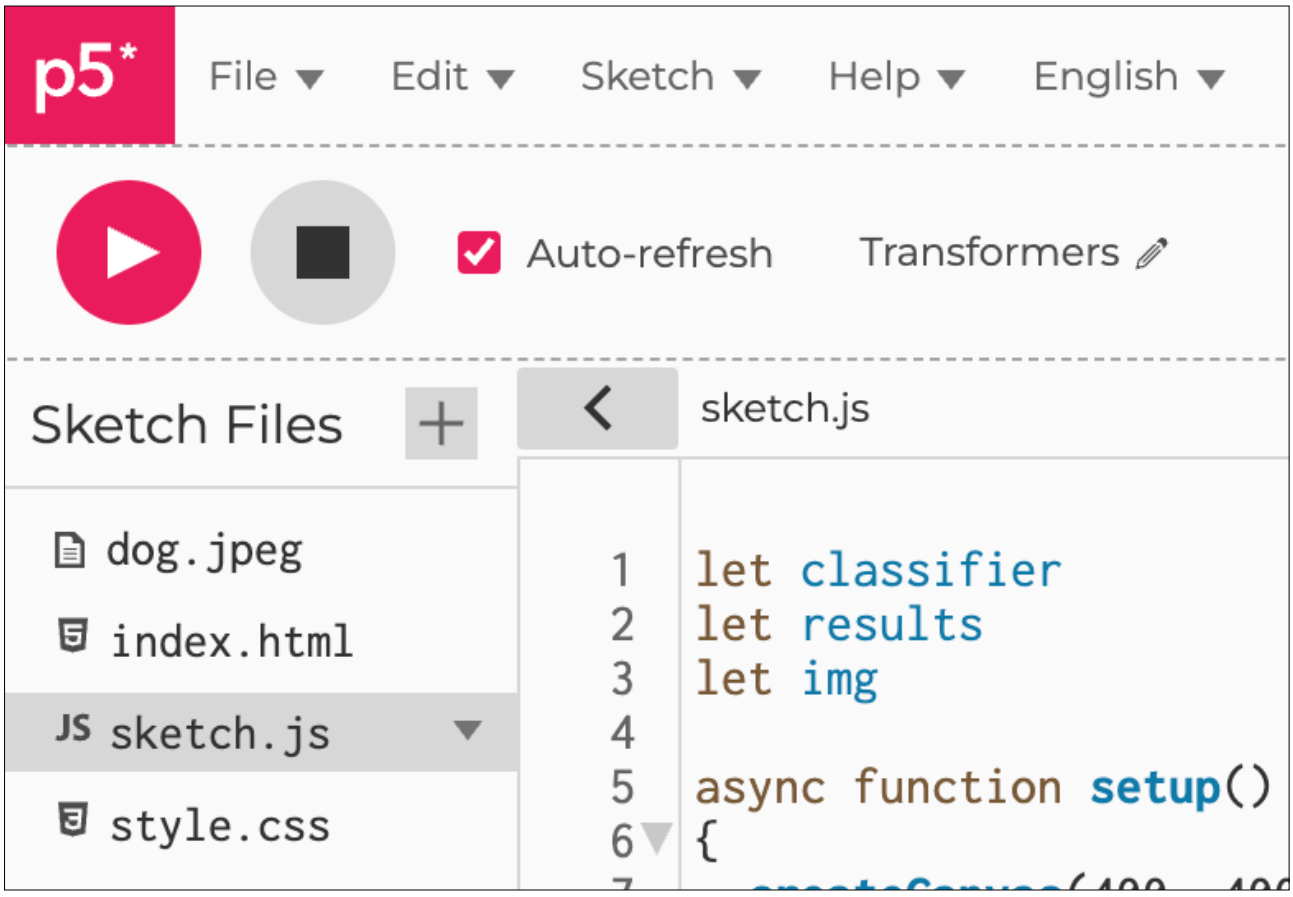


Figure 2: our canine friend

Figure 3: Sketch File (dog added)



Sketch C4.1 image detection

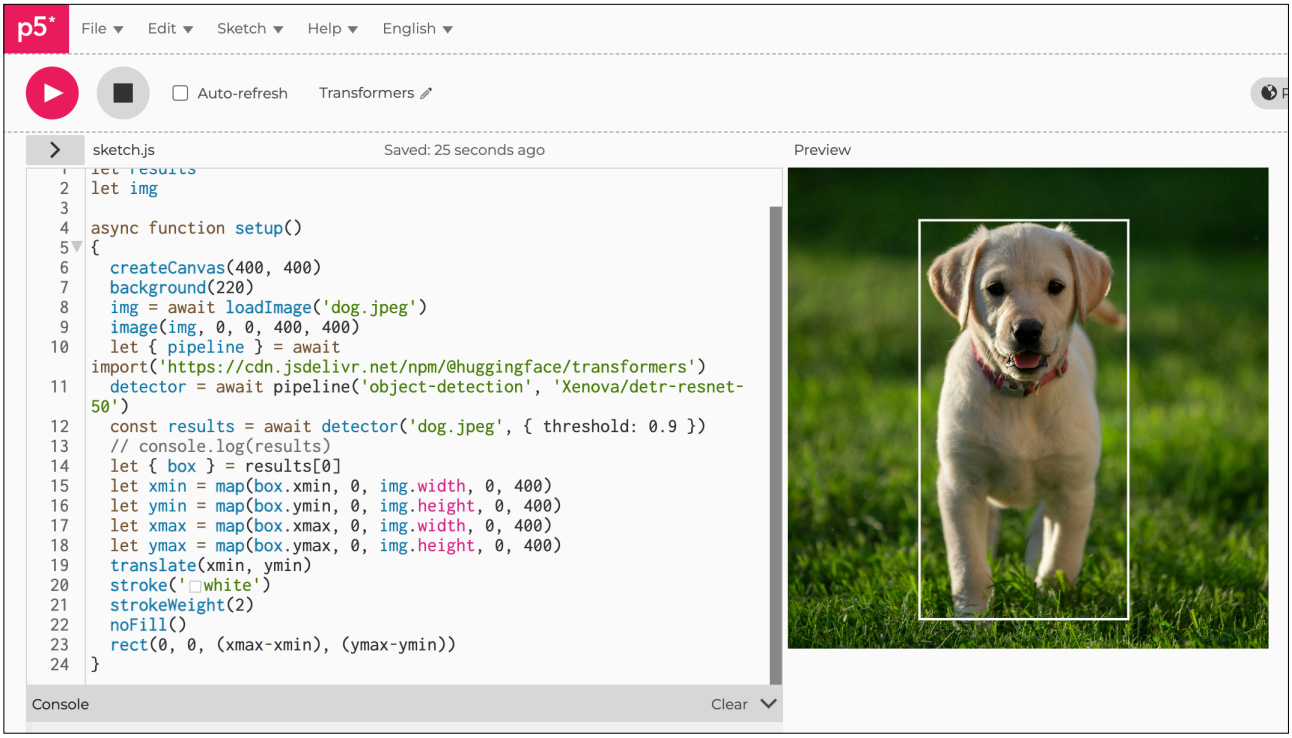
This puts a bounding box around an object it detects. In this case, it is the dog from the previous sketches.

```
let results
let img


async function setup()
{
  createCanvas(400, 400)
  background(220)
  img = await loadImage('dog.jpeg')
  image(img, 0, 0, 400, 400)
  let { pipeline } = await import('https://cdn.jsdelivr.net/npm/@huggingface/transformers')
  detector = await pipeline('object-detection', 'Xenova/detr-resnet-50')
  const results = await detector('dog.jpeg', { threshold: 0.9 })
  // console.log(results)
  let { box } = results[0]
  let xmin = map(box.xmin, 0, img.width, 0, 400)
  let ymin = map(box.ymin, 0, img.height, 0, 400)
  let xmax = map(box.xmax, 0, img.width, 0, 400)
  let ymax = map(box.ymax, 0, img.height, 0, 400)
  translate(xmin, ymin)
  stroke('white')
  strokeWeight(2)
  noFill()
  rect(0, 0, (xmax-xmin), (ymax-ymin))
}
```

Notes

We had to get the dimensions of the image on the canvas and map it to the real image size, which was considerably bigger. To see the raw results, uncomment the console.log.

Figure C4.1


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Transformers
sketch.js
Saved: 25 seconds ago
Preview
let img
async function setup()
{
createCanvas(400, 400)
background(220)
img = await loadImage('dog.jpeg')
image(img, 0, 0, 400, 400)
let { pipeline } = await
import('https://cdn.jsdelivr.net/npm/@huggingface/transformers')
detector = await pipeline('object-detection', 'Xenova/detr-resnet-
50')
const results = await detector('dog.jpeg', { threshold: 0.9 })
// console.log(results)
let { box } = results[0]
let xmin = map(box.xmin, 0, img.width, 0, 400)
let ymin = map(box.ymin, 0, img.height, 0, 400)
let xmax = map(box.xmax, 0, img.width, 0, 400)
let ymax = map(box.ymax, 0, img.height, 0, 400)
translate(xmin, ymin)
stroke('white')
strokeWeight(2)
noFill()
rect(0, 0, (xmax-xmin), (ymax-ymin))
}
Console
Clear

Sketch C4.2 adding the label

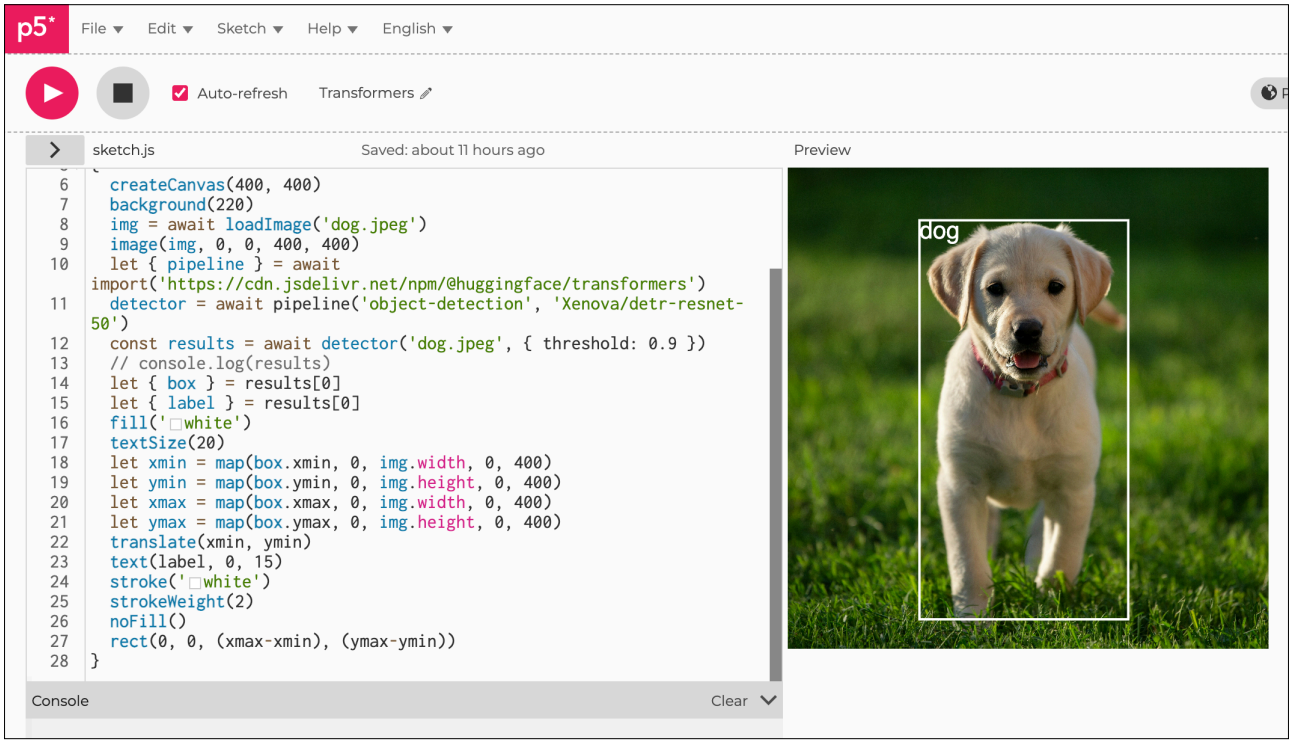
We can also create a label for the image that it has bounded. After all, it found an image of something it recognised.

```
let results
let img


async function setup()
{
  createCanvas(400, 400)
  background(220)
  img = await loadImage('dog.jpeg')
  image(img, 0, 0, 400, 400)
  let { pipeline } = await import('https://cdn.jsdelivr.net/npm/@huggingface/
transformers')
  detector = await pipeline('object-detection', 'Xenova/detr-resnet-50')
  const results = await detector('dog.jpeg', { threshold: 0.9 })
  // console.log(results)
  let { box } = results[0]
  let { label } = results[0]
  fill('white')
  textSize(20)
  let xmin = map(box.xmin, 0, img.width, 0, 400)
  let ymin = map(box.ymin, 0, img.height, 0, 400)
  let xmax = map(box.xmax, 0, img.width, 0, 400)
  let ymax = map(box.ymax, 0, img.height, 0, 400)
  translate(xmin, ymin)
  text(label, 0, 15)
  stroke('white')
  strokeWeight(2)
  noFill()
  rect(0, 0, (xmax-xmin), (ymax-ymin))
}
```

Notes

The label is useful to identify the object and if it is being used for training another model.

Figure C4.2


p5*
File
Edit
Sketch
Help
English
Auto-refresh
Transformers
sketch.js
Saved: about 11 hours ago
Preview
createCanvas(400, 400)
background(220)
img = await loadImage('dog.jpeg')
image(img, 0, 0, 400, 400)
let { pipeline } = await import('https://cdn.jsdelivr.net/npm/@huggingface/transformers')
detector = await pipeline('object-detection', 'Xenova/detr-resnet-50')
const results = await detector('dog.jpeg', { threshold: 0.9 })
// console.log(results)
let { box } = results[0]
let { label } = results[0]
fill('white')
textSize(20)
let xmin = map(box.xmin, 0, img.width, 0, 400)
let ymin = map(box.ymin, 0, img.height, 0, 400)
let xmax = map(box.xmax, 0, img.width, 0, 400)
let ymax = map(box.ymax, 0, img.height, 0, 400)
translate(xmin, ymin)
text(label, 0, 15)
stroke('white')
strokeWeight(2)
noFill()
rect(0, 0, (xmax-xmin), (ymax-ymin))
}
dog
Console
Clear

Sketch C4.3 multiple objects

Uploaded a new image with many objects, ran a `for()` loop to extract the data, although you have to be careful not to have too many objects, otherwise it will be cluttered. This is an example that pulls the dimensions of the image and resizes it on the canvas.

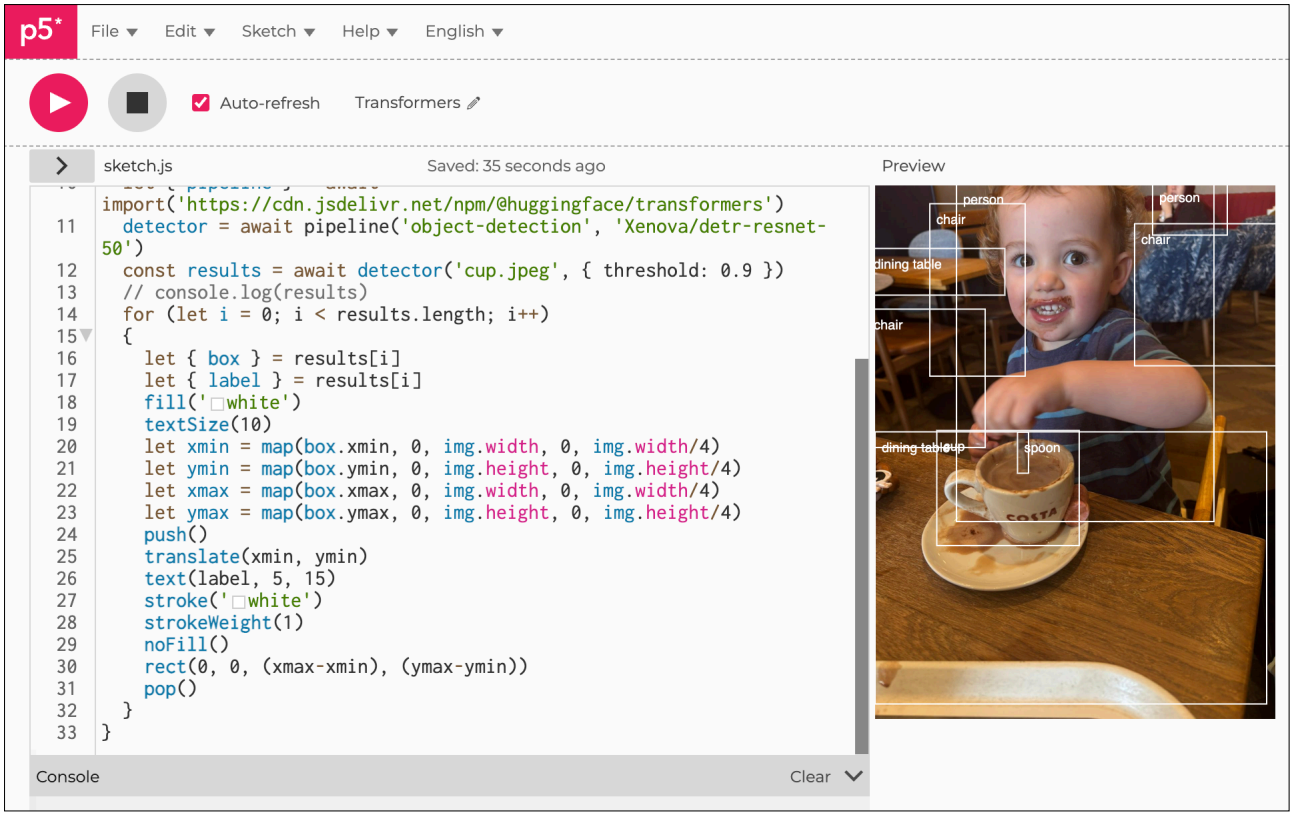
```
let results
let img


async function setup()
{
  createCanvas(600, 600)
  // background(220)
  img = await loadImage('cup.jpeg')
  image(img, 0, 0, img.width/4, img.height/4)
  let { pipeline } = await import('https://cdn.jsdelivr.net/npm/@huggingface/
transformers')
  detector = await pipeline('object-detection', 'Xenova/detr-resnet-50')
  const results = await detector('cup.jpeg', { threshold: 0.9 })
  // console.log(results)
  for (let i = 0; i < results.length; i++)
  {
    let { box } = results[i]
    let { label } = results[i]
    fill('white')
    textSize(10)
    let xmin = map(box.xmin, 0, img.width, 0, img.width/4)
    let ymin = map(box.ymin, 0, img.height, 0, img.height/4)
    let xmax = map(box.xmax, 0, img.width, 0, img.width/4)
    let ymax = map(box.ymax, 0, img.height, 0, img.height/4)
    push()
    translate(xmin, ymin)
    text(label, 5, 15)
    stroke('white')
    strokeWeight(1)
    noFill()
    rect(0, 0, (xmax-xmin), (ymax-ymin))
    pop()
```

```
  }
}
```

Notes

Resizing the image to fit on the canvas is a bit manual. I suspect it doesn't really matter if all you want is what objects are in an image (or video).

Figure C4.3



Quantisation

The weights are the stored parameters that a trained model uses when running the model for real. They can be stored as floating points, 32-bit, or 16-bit, for example. The more floating-point bit size, the more accurate it could be, but at the cost of being more computationally expensive.

This model works in your browser, so it needs to be as slimmed down as possible for it to work quickly and efficiently. One way is to have the weights stored as integers, and that is called quantisation.

In some models, you can specify this with such options as fp32 or q4, for instance. This is what these values mean. By default, the model may store weights as 32-bit floating points. The weird thing is that by reducing the number of floating-point decimal places, you don't lose much accuracy and you gain a faster result.