

# Algorithmic Intelligence

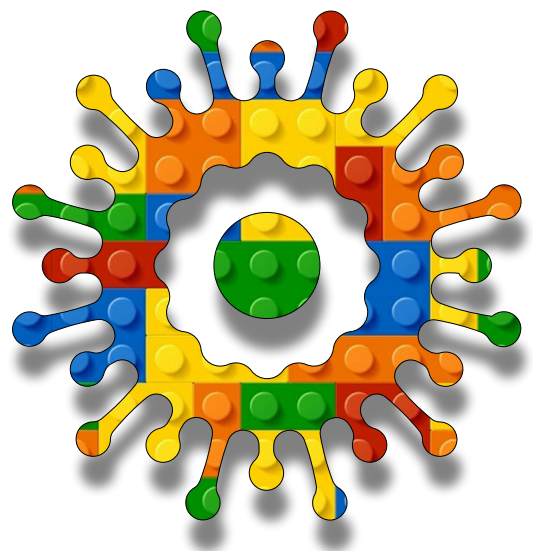
## Module C

### Unit #2

#### transformer

#### image

#### classifier





## Module C Unit #2 transformer image classifier

Choosing the task

The task tab

Select one

Their example

Upload the dog

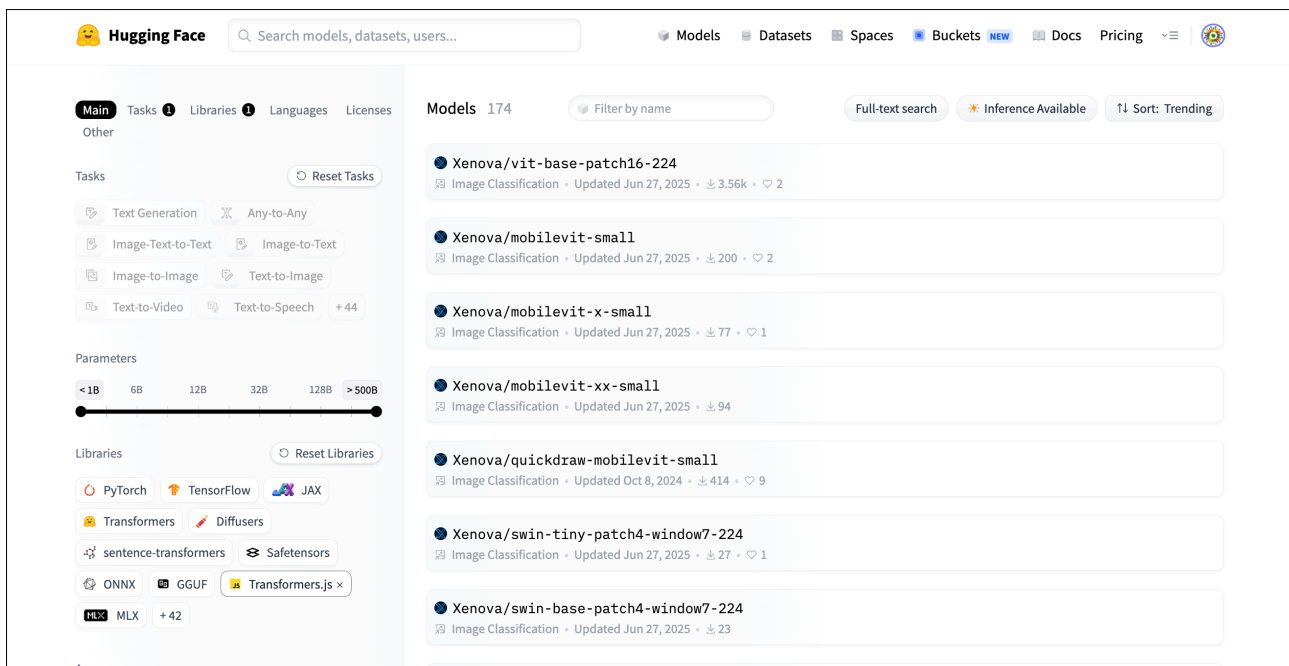
Sketch C2.1      the code



# Introduction to transformer image classification

In this example from [Hugging Face](#), we make use of an image classifier. First, we select `transformers.js` from the main tab.

Figure 1: main tab - transformers.js

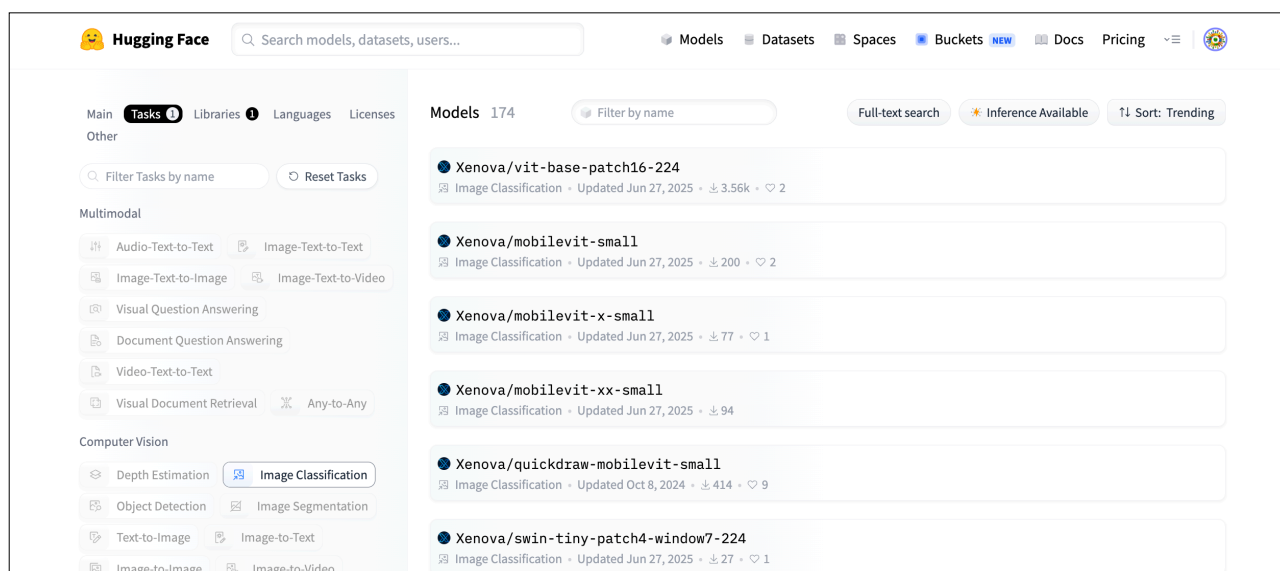




# Choosing the task

From the **task** tab, we select **image classification**.

Figure 2: image classifier task

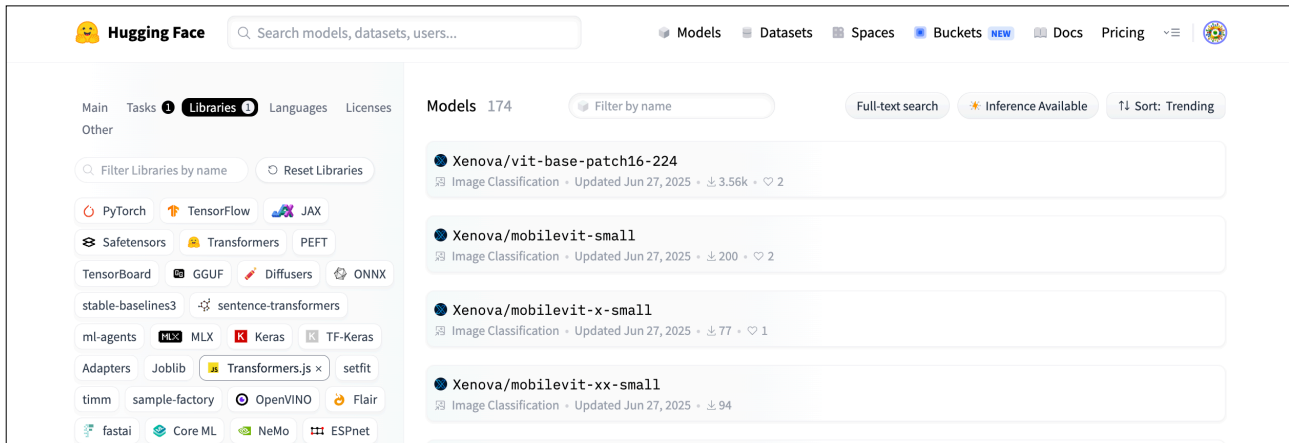




# The task tab

In case you didn't select the library, we can do that from the library tab and select **transformers.js**.

Figure 3: task tab keck



## Select one

We select the top one, and we get the web page that corresponds to it.

Figure 4: corresponding web page

The screenshot shows the Hugging Face model page for 'Xenova/vit-base-patch16-224'. The page includes a search bar, navigation tabs (Models, Datasets, Spaces, Buckets, Docs, Pricing), and a 'Use this model' button. The main content area displays the model's name, tags (Image Classification, Transformers.js, ONNX, vit), and a 'Model card' tab. The model card text includes a URL, usage instructions for Transformers.js, and an example code snippet. The right sidebar shows 'Downloads last month' (3,564), 'Inference Providers' (Image Classification), 'Model tree for Xenova/vit-base-patch16-224' (Base model: google/vit-base-patch16-224, Quantized: 11), and 'Spaces using Xenova/vit-base-patch16-224' (11).

We want some help in getting started. First off, we click on the **use this model** tab, which will ask us to use the **transformers.js** library. Click on that, and you get the following information, which is what we need to insert into our code.

```
import { pipeline } from '@huggingface/transformers';
```

```
const pipe = await pipeline('image-classification', 'Xenova/vit-base-patch16-224');
```

The bit we want is highlighted in blue. We put it inside the speech marks inside the pipeline as before.



## Their example

However, that is not all; we want to use an image. To help us, we can look at their example on the left-hand side of the page. We can copy it as such:

```
import { pipeline } from '@huggingface/transformers';

const classifier = await pipeline('image-classification', 'Xenova/
vit-base-patch16-224');
const urls = [
  'https://huggingface.co/datasets/Xenova/transformers.js-docs/
resolve/main/tiger.jpg',
  'https://huggingface.co/datasets/Xenova/transformers.js-docs/
resolve/main/cats.jpg',
];
const output = await classifier(urls);
// [
//   { label: 'tiger, Panthera tigris', score:
0.6074584722518921 },
//   { label: 'Egyptian cat', score: 0.8246098756790161 }
// ]
```

The bit that we are interested in is highlighted in blue. Instead of a URL, we can upload an image and call that image directly. It will load the image and classify it. We will also display it on the canvas.



## Upload the dog

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

Figure 5: download dog image

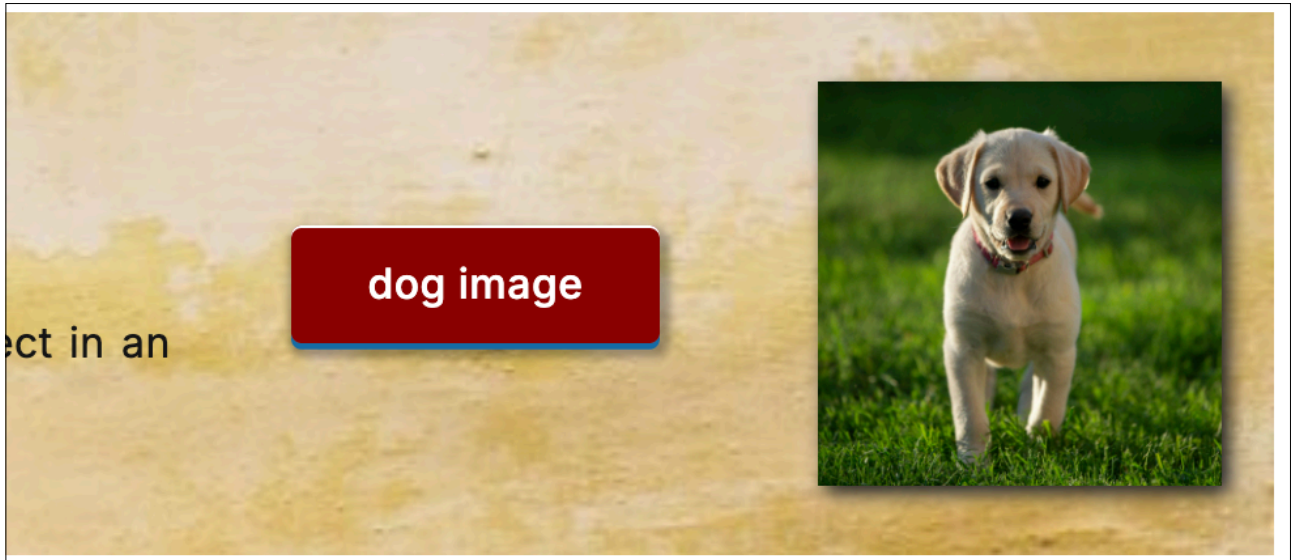


Figure 6: our canine friend

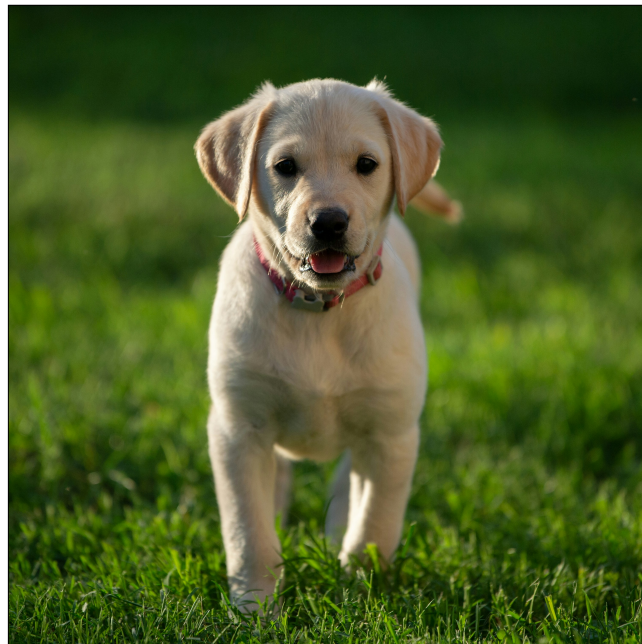
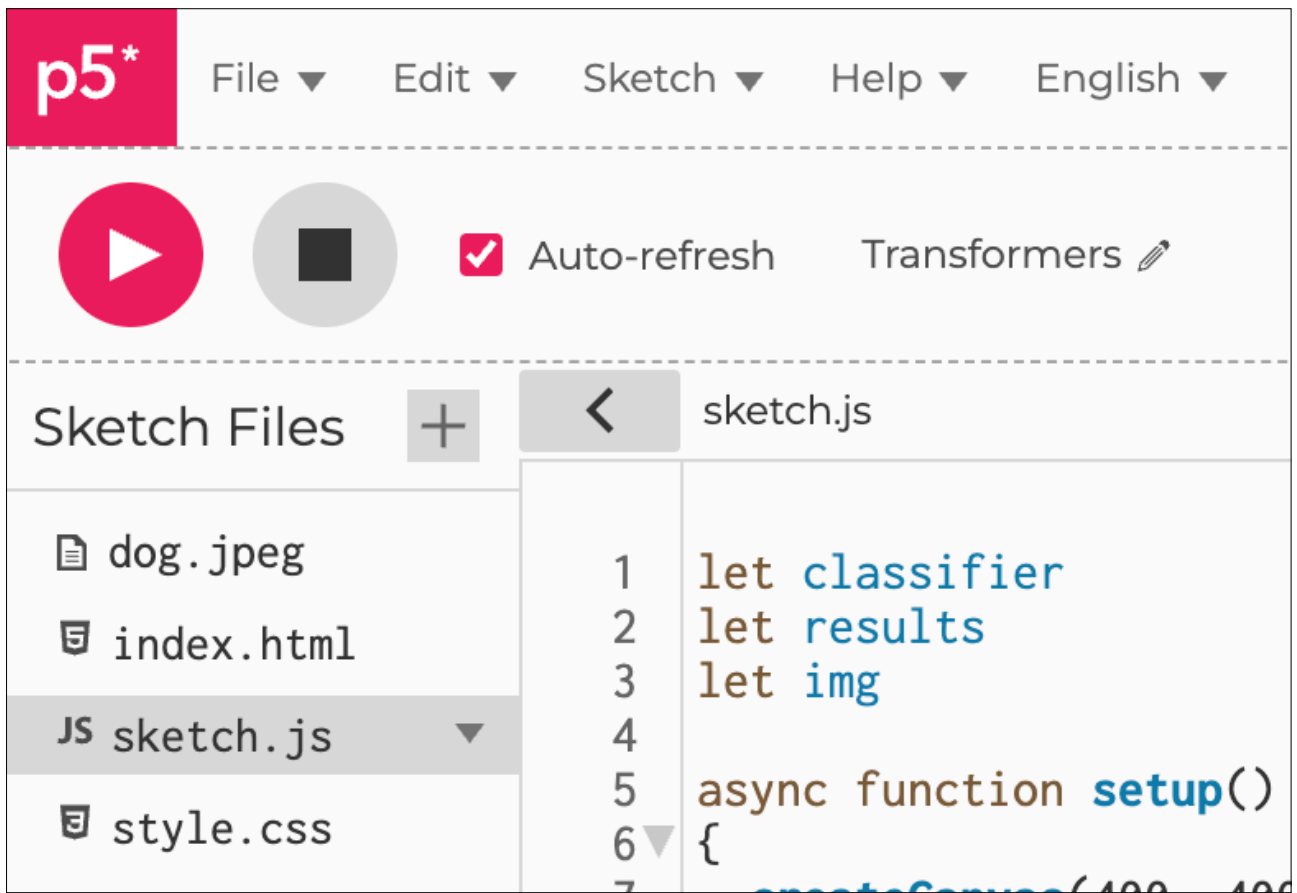


Figure 7: Sketch File (dog added)





## Sketch C2.1 the code

This is virtually the same code.

! Remove reference to the **button** and the **text box** and **text**.

```
let classifier
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')

  classifier = await pipeline('image-classification', 'Xenova/vit-base-patch16-224')
  results = await classifier('dog.jpeg')
  fill('white')

  let { label, score } = results[0]
  textSize(20)

  text(label, 10, 350)
  text(floor(score * 100) + '%', 10, 380)
}
```



### Notes

You can see from the results that we have a good match.



### Challenge

Try other animals or objects.

Figure C2.1

The image shows a p5.js IDE interface. The top bar includes the p5.js logo, a menu (File, Edit, Sketch, Help, English), and a user greeting 'Hello, ElegantAI!'. Below the menu are buttons for 'Auto-refresh' (checked) and 'Transformers'. On the right, there are buttons for 'Public', 'p5.js 2.2.2', and a settings gear icon. The main workspace is split into two panes: a code editor on the left and a preview window on the right. The code editor shows a JavaScript file named 'sketch.js' with the following code:

```
1 let classifier
2 let results
3 let img
4
5 async function setup()
6 {
7   createCanvas(400, 400)
8   background(220)
9   img = await loadImage('dog.jpeg')
10  image(img, 0, 0, 400, 400)
11  let { pipeline } = await import('https://cdn.jsdelivr.net/npm/@huggingface/transformers')
12  classifier = await pipeline('image-classification', 'Xenova/vit-base-patch16-224')
13  results = await classifier('dog.jpeg')
14  fill('white')
15  let { label, score } = results[0]
16  textSize(20)
17  text(label, 10, 350)
18  text(floor(score * 100) + '%', 10, 380)
19 }
```

The preview window shows a photograph of a light-colored Labrador retriever puppy standing in a grassy field. Overlaid on the bottom left of the image is the text 'Labrador retriever' and '97%'. At the bottom of the IDE, there is a 'Console' pane with a 'Clear' button.