Image Classification using Javascript

IDC-2(SEM-VIII)

Saket Kumar Sahu(17086)

May 4, 2021

Contents

Introduction

In this project I have made a website which can classify various the images uploaded by the user. The website is created using HTML, CSS and Javascript. The model is trained on data from ImageNet, which is an image database organized according to the WordNet hierarchy (currently only the nouns), in which each node of the hierarchy is depicted by hundreds and thousands of images. The project has been instrumental in advancing computer vision and deep learning research. The data is available for free to researchers for non-commercial use.

The trained model is used through Javascript using the ml5 project. The ml5 project is about making machine learning approachable to a broader audience. This pretrained model used in this project is “MobileNet” which is a term used for a type of machine learning model architecture that has been optimised to run on platforms with limited computational power. They are used for various purposes like image classification, image segmentation and object detection. The MobileNet model used here was trained for image classification.

If one opens the HTML file in some text editor(here I used Visual Studio Code) we can see that we have only the index.html file of the website. There is no separate Javascript file required to run the webpage. This is because the javascript file is taken from the cloud as a result of which an active internet connection is required to run the webpage. The ml5.js project is based on tensorflow.js and we see that it is possible to use the web browser’s built in graphics processing unit (GPU) to do calculations that would otherwise run very slowly using central processing unit (CPU).

ml5 uses a MobileNet created with TensorFlow.js, a JavaScript library from TensorFlow. There are several TensorFlow.js MobileNet versions available for image classification, and as of June 2019, ml5 defaults to importing MobileNetV2. This trained model is developed by Google’s Tensorflow.js team and it is trained on a database of approximately 15 million images.

If we want to use our own trained model we can use it as well. We have to provide the url to the model json file in the html code. Google’s Teachable Machine(`https://teachablemachine.withgoogle.com/train`) makes this task much more easier and simpler by giving us the trained model directly according to the different classes which we need. It directly takes different input images from the user and uses to train the model. It trains the model by doing transfer learning. It is built on top of TensorFlow.js(a library for machine learning in

Javascript, to train and run the models you make in your web browser). There's a pretrained neural network, and when one creates one's own classes,the classes become the last layer or step of the neural net. Specifically, both the image and pose models are learning off of pretrained mobilenet models. For this reason it can train the models very quickly which would have taken much more time if done on one's own CPU.

Analysing the code

In this section we will go through some snippets of the html code and try to understand what is happening underneath

```
<title>Image Classifier using ML5 js</title>
```

Here we have defined the title of the webpage which appears as the tab in the browser window.

```
<script src="https://unpkg.com/ml5@0.4.3/dist/ml5.min.js">
</script>
```

This one line of code is basically the java script file for the model. So an active connection is required to open this link (https://unpkg.com/ml5@0.6.1/dist/ml5.min.js). On opening this link one will be able to see the details present in the trained model which will be in json format. This is basically is what is doing all the work and rest all parts in the code is just helping this model out.

```
<script>
var loadFile = function (event){
var image = document.getElementById("image");
image.src = URL.createObjectURL(event.target.files[0]);
};
const classifier = ml5.imageClassifier('MobileNet', modelLoaded);
// When the model is loaded
function modelLoaded(){
console.log("Model␣Loaded!");
}
function predict(){
classifier.predict(document.getElementById("image"),
function (err, results){
alert(results[0].label);
});
}
</script>
```

This part of the javascript code is responsible for the functionality of the webpage. Here we are creating a variable named loadFile and image, loadFile is basically a function but it is an anonymous function which contains all the information about the event which is being passed to it. In our case this event is the user uploading the image to the website and we use this information to react to this event like we are storing the URL of the uploaded image in our defined image variable. This allows us to use our model on the image variable which we will see later.

We are defining a constant classifier which is a “MobileNet” model. The other argument in classifier is a function to register in the console that the model is loaded.This part is not visible to the user and can be accessed using the developer tools of the browser. After this we predict what is the image by using the predict function present in our trained classifier. The output is given as an alert in the website.

Whatever we have seen till now is in the head part of the HTML code. Now we will see the body which is about styling of the website which is handled by CSS.

```
<body>
<center>
<h1 style="color: rgb(0, 60, 128);">
Saket Image Classifier</h1>
<b>
Image Classification using Javascript
</b>
<p>Upload the Picture which you want to classify</p>
</br>
<img src="" alt="" id="image"
width="315px" height="200px" />
</br></br>
<input type="file" accept="image/*"
onchange="loadFile(event)"
name="image" id="file" />
<button onclick="predict()">Predict</button>
</center>
</body>
```

Here we see the heading which is of h1 type and after that are the information for the user. In the next part we are resizing the image to 315px by 200px since our model is trained accordingly. The br elements are the linebreaks. The user is only allowed to input image files which is evident from the accept part of the input section. After that a button is created and the predict function is assigned to it, clicking which we will get the type of classification as an alert in the webpage.

Steps to run the website

- Make sure you have an active internet connection and you have the image you want to predict downloaded as .jpeg/.jpg/.png format.
- Host a local server
- Run the index.html file
- If you want to view the code in the html run it using a debugger of a text editor or the developer tools of the web browser.

Results and Conclusion

Here are some of the classifications made using this model in the website

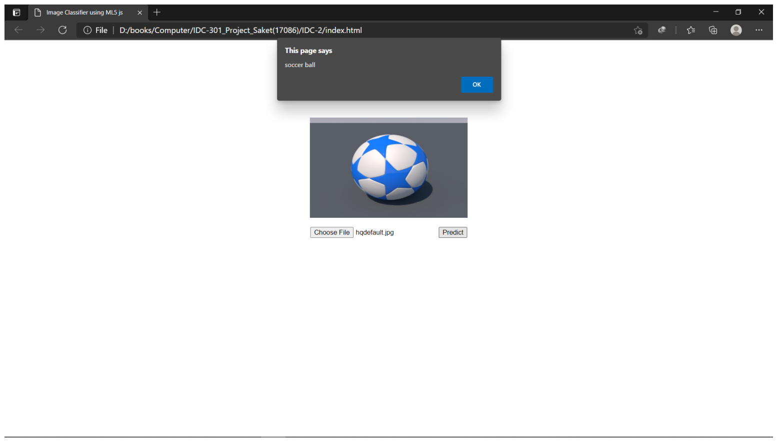


Image Classifier using ML5 js
File | D:/books/Computer/IDC-301_Project_Saket(17086)/IDC-2/index.html
OK
Choose File
hqdefault.jpg
Predict

This page says

soccer ball

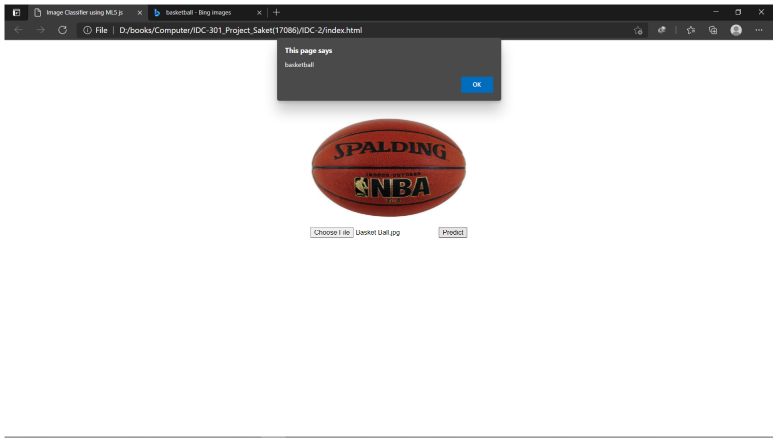
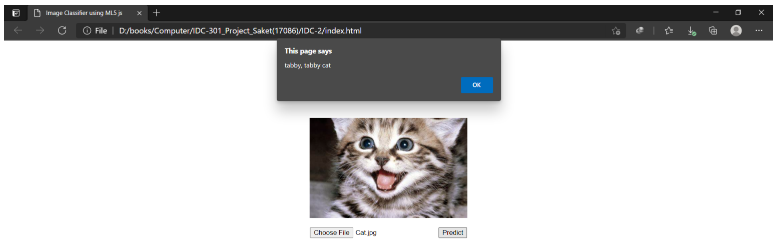


Image Classifier using ML5 js
basketball - Bing images
File | D:/books/Computer/IDC-301_Project_Saket(17086)/IDC-2/index.html
OK
SPALDING
NBA
Choose File
Basket Ball.jpg
Predict

This page says

basketball



D:/books/Computer/IDC-301_Project_Saket(17086)/IDC-2/index.html

This page says

tabby, tabby cat

As we can see it has classified everything correctly. It even has identified the breed of the cat. Thus this kind of models are now being used extensively in many places. We can also use our own trained model by just replacing the MobileNet part in the classifier with the path to our model. The underlying principle behind all this models is still the concept of convulutional neural networks which play a crucial part in Deep Learning.

Bibliography

[1] https://learn.ml5js.org/#/reference/image-classifier

[2] https://www.image-net.org/

[3] https://en.wikipedia.org/wiki/Main_Page