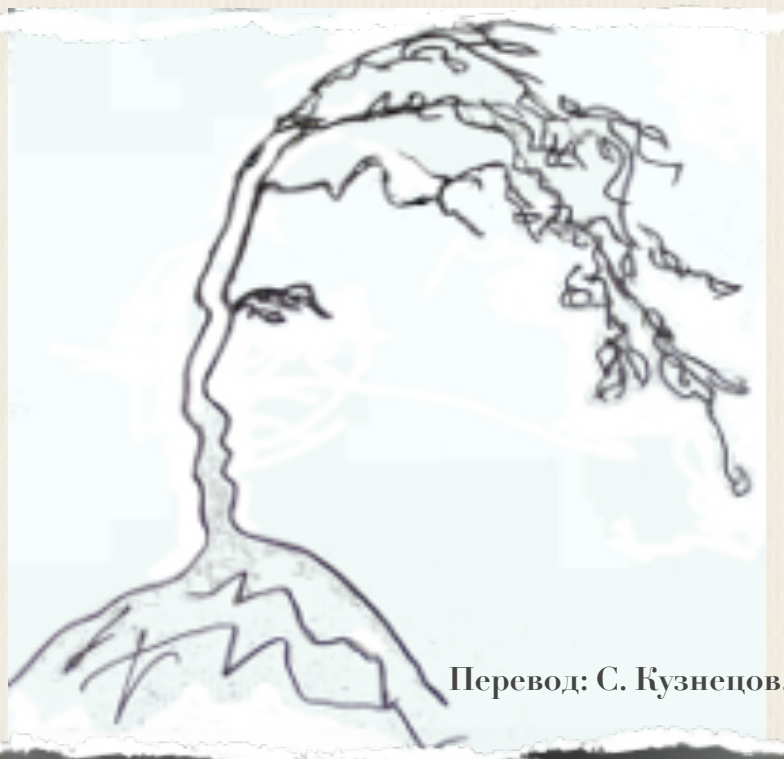
РАФАЭЛЬ МУН(RAPHAEL MUN)

Статьи по машинному обучению в браузере с использованием фреймворка TensorFlow.js

УЧЕБНЫЕ РУКОВОДСТВА



Перевод: С. Кузнецов, 2023 г.

Articles on machine training in the browser with use of a framework TensorFlow.js

Raphael Mun

2020 · 2001

https://www.codeproject.com/Articles/instafluff#Article

Статьи по машинному обучению в браузере с использованием фреймворка TensorFlow.js

Рафаэль Мун

2020 · 2001

https://www.codeproject.com/Articles/instafluff#Article

Перевод: С. Кузнецов, 31.10.2023

Статья 2 «Собаки и пицца: машинное зрение в браузере с использованием TensorFlow.js»

Статья 2 **«Собаки и пицца: машинное зрение в браузере с использованием TensorFlow.js»(«Dogs and Pizza: Computer Vision in the Browser With TensorFlow.js»;** https://www.codeproject.com/Articles/5272771/Dogs-and-Pizza-Computer-Vision-in-the-Browser-With) является статьей из серии статей **Обнаружение касания лица с помощью Tensorflow.js(Face Touch Detection with Tensorflow.js)**

В этой статье мы погрузимся в **машинное зрение(computer vision)** прямо внутри веб-браузера.

Здесь мы рассмотрим темы:

- сбор тестовых изображений
- **TensorFlow.js**-импорты и коллекция демонстрационных изображений
- добавление элемента изображения (установлен в **224** пкс)
- добавление элемента элемента текста состояния с функциями установки изображений
- добавление кода для выборки в произвольном порядке и загрузки одного из изображений
- выполнение распознавания объекта
- добавление кода для показа результата прогноза в тексте

TensorFlow + JavaScript. Самый популярный, ультрасовременный **AI**-фреймворк(инфраструктура) теперь поддерживает наиболее широко используемый язык программирования на планете, поэтому давайте

заставим волшебство произойти посредством **глубокого изучения(deep learning)** прямо в нашем веб-браузере, ускоренном **графическим процессорным устройством(ГПУ; GPU)** через графическую библиотеку **WebGL**, используя фреймворк машинного обучения **TensorFlow.js**!

В этой статье мы погрузимся в **машинное зрение(computer vision)** прямо внутри веб-браузера. Я покажу вам, как легко и быстро вы можете начать **обнаруживать и распознавать объекты(detecting and recognizing objects)** в изображениях, используя в фреймворке машинного обучения **TensorFlow.js** предварительно обученную MobileNet-модель.

Maltese dog

Стартовая точка

Для анализа изображений с использованием фреймворка машинного обучения **TensorFlow.js**, мы сначала должны:

- Собрать тестовые изображения собак, еды и многого другого(изображения, используемые в этом проекте, получены с сайта pexels.com)
- Импортировать фреймворк машинного обучения **TensorFlow.js**

- Добавить элемент изображения(установлен в `224` пкс)
- Добавить код для выборки в произвольном порядке и загрузки одного из изображений
- Добавить код для показа результата прогноза в тексте

Вот наша стартовая точка:

JavaScript

```
<html>
    <head>
        <title>Dogs and Pizza: Computer Vision in the Browser with
TensorFlow.js</title>
        <script
src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@2.0.0/dist/tf.min.js"><
/script>
        <style>
            img {
                object-fit: cover;
            }
        </style>
    </head>
    <body>
        <img id="image" src="" width="224" height="224" />
        <h1 id="status">Загрузка.../Loading...</h1>
        <script>
        const images = [
             "web/dalmation.jpg", //
https://www.pexels.com/photo/selective-focus-photography-of-woman-holding-
adult-dalmatian-dog-1852225/
            "web/maltese.jpg", //
https://www.pexels.com/photo/white-long-cot-puppy-on-lap-167085/
            "web/pug.jpg", //
https://www.pexels.com/photo/a-wrinkly-pug-sitting-in-a-wooden-table-34756
80/
            "web/pomeranians.jpg", //
https://www.pexels.com/photo/photo-of-pomeranian-puppies-4065609/
            "web/pizzaslice.jpg", //
https://www.pexels.com/photo/pizza-on-brown-wooden-board-825661/
            "web/pizzaboard.jpg", //
https://www.pexels.com/photo/pizza-on-brown-wooden-board-825661/
            "web/squarepizza.jpg", //
https://www.pexels.com/photo/pizza-with-bacon-toppings-1435900/
            "web/pizza.jpg", //
https://www.pexels.com/photo/pizza-on-plate-with-slicer-and-fork-2260200/
            "web/salad.jpg", //
https://www.pexels.com/photo/vegetable-salad-on-plate-1059905/
```

```
            "web/salad2.jpg", //
https://www.pexels.com/photo/vegetable-salad-with-wheat-bread-on-the-side-
1213710/
            "web/kitty.jpg", //
https://www.pexels.com/photo/eyes-cat-coach-sofa-96938/
            "web/upsidedowncat.jpg", //
https://www.pexels.com/photo/silver-tabby-cat-1276553/
        ];

        function pickImage() {
            document.getElementById( "image" ).src = images[ Math.floor(
Math.random() * images.length ) ];
        }

        function setText( text ) {
            document.getElementById( "status" ).innerText = text;
        }

        (async () => {

            // Ваш код поместите сюда
            // Your Code Goes Here

            setInterval( pickImage, 5000 );
        })();
        </script>
    </body>
</html>
```

Этот веб-код импортирует фреймворк машинного обучения `TensorFlow.js` и набор демонстрационных изображений, добавляет элемент изображения (установлен в `224` пкс), и затем добавляет текстовый элемент состояния с функциями, чтобы установить изображения. Вы можете обновить массив изображений, чтобы соответствовать, именам файлов других демонстрационных изображений, которые вы вас загрузите.

В браузере откройте страницу с вышеупомянутым кодом. Вы должны видеть, что изображение обновляется каждые пять секунд по случайному выбору.

Примечание по хостингу

Прежде чем мы пойдем далее, я хочу отметить, чтобы этот проект выполнялся правильно, веб-страница и изображения должны быть размещены на веб-сервере (из-за `canvas`-ограничений языка `HTML5`).

Иначе вы можете столкнуться с этой ошибкой, когда фреймворк машинного обучения `TensorFlow.js` пытается считать пиксели изображения:

```
DOMException: Failed to execute 'texImage2D' on 'WebGL2RenderingContext':
Tainted canvases may not be loaded.
```

Если у вас нет сервера выполнения подобного `Apache`-серверу или `IIS`-серверу, то вы можете выполнить тот легко в продукте менеджера узла модулей `NodeJS` с веб-сервером-модулем WebWebWeb, `NPM`-модулем, который я построил для этой самой цели.

Обученная модель MobileNet v1

С сервиса `GitHub`: `"MobileNets небольшие модели, с низкой задержкой, маломощные модели, параметризованные, для множества вариантов использования при встречаемых ограничениях ресурсов"("MobileNets are small, low-latency, low-power models parameterized to meet the resource constraints of a variety of use cases.")`

Для этого проекта мы будем использовать модель `MobileNet v1`, которая была обучена на миллионах изображений для распознавания `1000` различных категорий объектов(1000 different categories of objects) в диапазоне от различных пород собак до различных типов еды. Есть множество вариаций модели, чтобы позволить разработчикам делать компромиссы между сложностью/размером и точностью прогноза.

К счастью корпорация `Google` размещает модель на их серверах и поэтому мы можем использовать ее напрямую в нашем проекте. Мы будем использовать самое маленькое, вариации `0.25` патча для ввода изображения на `224x224` пикселя.

Давайте добавим ее к нашему скрипту и загрузим модель с помощью фреймворк машинного обучения `TensorFlow.js`.

JavaScript

```
// Mobilenet v1 0.25 224x224 model
```

```
const mobilenet =
"https://storage.googleapis.com/tfjs-models/tfjs/mobilenet_v1_0.25_224/mod
el.json";

let model = null;

(async () => {
    // Загрузка модели
    // Load the model
    model = await tf.loadLayersModel( mobilenet );
    setInterval( pickImage, 5000 );
})();
```

Выполнение распознавания объекта

Прежде, чем `TensorFlow`-модели передать наше изображение, мы должны **`преобразовать его из пикселей в тензоры(convert it from pixels to tensors)`**. Мы также должны `RGB`-цвет каждого пикселя преобразовать в значение с плавающей точкой в диапазоне между `-1.0` и `1.0`, потому что это - диапазон значений, на котором была обучена `MobileNet`-модель.

У фреймворка `TensorFlow.js` есть встроенные функции, чтобы помочь нам легко выполнять эти операции. Имея в виду управление памятью, мы можем записать функцию выполнения модели и получить предсказанный идентификатор объекта, как в этом коде:

JavaScript

```
async function predictImage() {
    let result = tf.tidy( () => {
        const img = tf.browser.fromPixels( document.getElementById(
"image" ) ).toFloat();
        const normalized = img.div( 127 ).sub( 1 ); //
                // Normalize from [0,255] to [-1,1]
                // Нормализуем из диапазона [0,255] в диапазон [-1,1]
        const input = normalized.reshape( [ 1, 224, 224, 3 ] );
        return model.predict( input );
    });
    let prediction = await result.data();
    result.dispose();
    // Получим индекс самого высокого значения в прогнозе
    // Get the index of the highest value in the prediction
    let id = prediction.indexOf( Math.max( ...prediction ) );
    setText( labels[ id ] );
```

```
}
```

Вы заметите, что вышеупомянутая функция ссылается на массив меток `labels`, чтобы показать предсказанный текст с помощью функции `setText()`. Этот массив - предопределенный `список подписей категорий ImageNet` (ImageNet category labels), которые совпадают с прогнозами, на которых модель `MobileNet` была обучена.

Если вы загружаете это, не забудьте включать его в теге скрипта `<script>` на вашей странице в теге `<head>`, как в этом отрывке кода:

JavaScript

```
<script src="web/labels.js"></script>
```

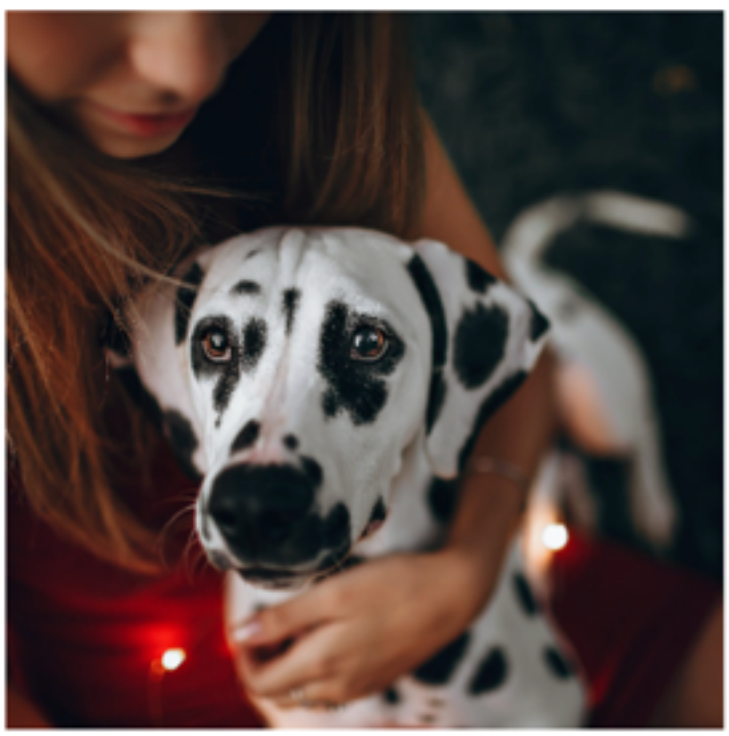
Наконец, давайте на элементе изображения сделаем вызов выполнения этой функции прогноза каждый раз, когда загружено новое изображение, установив его обработчик загрузки `onload` внизу нашего кода:

JavaScript

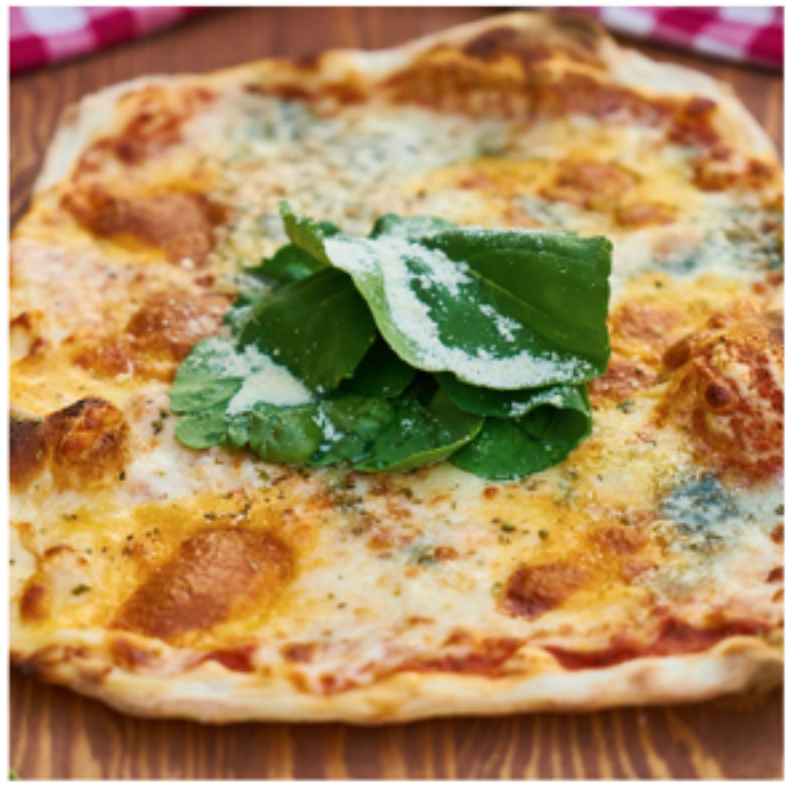
```
(async () => {
    // Загрузка модели
    // Load the model
    model = await tf.loadLayersModel( mobilenet );
    setInterval( pickImage, 5000 );
    document.getElementById( "image" ).onload = predictImage;
})();
```

Как только все сделано полностью, ваша модель должна начать предсказывать то, что находится в изображении.

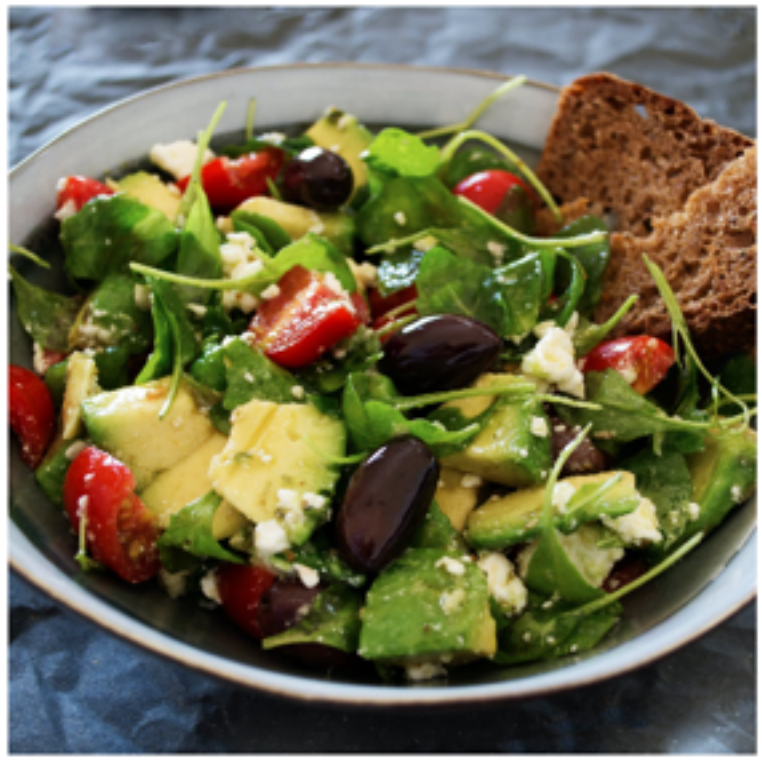
Pomeranian



dalmatian



pizza



guacamole

Финишная черта

Теперь вы вместе объединили все части. Точно так же, как эта, у вас есть веб-страница, использующая **машинное зрение и может распознать объекты(computer vision and can recognize objects)**. Не плохо, а?

JavaScript

```
<html>
    <head>
        <title>Dogs and Pizza: Computer Vision in the Browser with
TensorFlow.js</title>
        <script
src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@2.0.0/dist/tf.min.js"><
/script>
         <style>
             img {
                 object-fit: cover;
             }
        </style>
        <script src="web/labels.js"></script>
    </head>
    <body>
        <img id="image" src="" width="224" height="224" />
        <h1 id="status">Загрузка/Loading...</h1>
        <script>
        const images = [
            "web/dalmation.jpg", //
https://www.pexels.com/photo/selective-focus-photography-of-woman-holding-
adult-dalmatian-dog-1852225/
            "web/maltese.jpg", //
https://www.pexels.com/photo/white-long-cot-puppy-on-lap-167085/
            "web/pug.jpg", //
https://www.pexels.com/photo/a-wrinkly-pug-sitting-in-a-wooden-table-34756
80/
            "web/pomeranians.jpg", //
https://www.pexels.com/photo/photo-of-pomeranian-puppies-4065609/
            "web/pizzaslice.jpg", //
https://www.pexels.com/photo/pizza-on-brown-wooden-board-825661/
            "web/pizzaboard.jpg", //
https://www.pexels.com/photo/pizza-on-brown-wooden-board-825661/
            "web/squarepizza.jpg", //
https://www.pexels.com/photo/pizza-with-bacon-toppings-1435900/
            "web/pizza.jpg", //
https://www.pexels.com/photo/pizza-on-plate-with-slicer-and-fork-2260200/
            "web/salad.jpg", //
https://www.pexels.com/photo/vegetable-salad-on-plate-1059905/
```

```
            "web/salad2.jpg", //
https://www.pexels.com/photo/vegetable-salad-with-wheat-bread-on-the-side-
1213710/
            "web/kitty.jpg", //
https://www.pexels.com/photo/eyes-cat-coach-sofa-96938/
            "web/upsidedowncat.jpg", //
https://www.pexels.com/photo/silver-tabby-cat-1276553/
        ];

        function pickImage() {
            document.getElementById( "image" ).src = images[ Math.floor(
Math.random() * images.length ) ];
        }

        function setText( text ) {
            document.getElementById( "status" ).innerText = text;
        }

        async function predictImage() {
            let result = tf.tidy( () => {
                const img = tf.browser.fromPixels(
document.getElementById( "image" ) ).toFloat();
                const normalized = img.div( 127 ).sub( 1 ); // Normalize
from [0,255] to [-1,1]
                const input = normalized.reshape( [ 1, 224, 224, 3 ] );
                return model.predict( input );
            });
            let prediction = await result.data();
            result.dispose();
            // Get the index of the highest value in the prediction
            let id = prediction.indexOf( Math.max( ...prediction ) );
            setText( labels[ id ] );
        }

        // Mobilenet v1 0.25 224x224 model
        const mobilenet =
"https://storage.googleapis.com/tfjs-models/tfjs/mobilenet_v1_0.25_224/mod
el.json";

        let model = null;

        (async () => {
            // Load the model
            model = await tf.loadLayersModel( mobilenet );
            setInterval( pickImage, 5000 );
            document.getElementById( "image" ).onload = predictImage;
        })();
        </script>
    </body>
</html>
```

Что далее? Пушистые животные?

Просто с небольшим отрывком кода мы загрузили **предварительно обученную модель(pre-trained model)**, используя фреймворк машинного обучения **TensorFlow.js**, чтобы **распознать объекты(recognize objects)** из списка в веб-браузере. Вообразите то, что вы могли сделать с этим. Возможно, создать веб-приложение, которое автоматически классифицирует и сортирует тысячи фотографий.

Теперь, а что, если мы захотим распознать другие объекты, которые не находятся в списке **1000** категорий?

В серии статей, смотрите следующую статью **Детектор пушистых животных: распознавание нестандартных объектов в браузере, путем передачи обученности в фреймворке TensorFlow.js(Fluffy Animal Detector: Recognizing Custom Objects in the Browser by Transfer Learning in TensorFlow.js)**.

Вы изучите, как можно развернуть этот проект для быстрого **обучения(тренировки) сверточной нейронной сети(Convolutional Neural Network)** для **распознавания...(recognize...)** чего-либо желаемого.

So Cute & Fluffy!

Эта статья - статья из серии статей **Обнаружение касания лица с помощью Tensorflow.js(Face Touch Detection with Tensorflow.js)**