Tutorial 1

Clasificación de imágenes con MobileNet

Introducción

En este tutorial aprenderás cómo cargar y ejecutar un popular modelo previamente entrenado, llamado **MobileNet**, para la clasificación de imágenes en el navegador. El repositorio del modelo MobileNet está disponible en la siguiente dirección: https://github.com/tensorflow/tfjs-models/tree/master/mobilenet. MobileNet utiliza la base de datos **ILSVRC** de **ImageNet**, con más de 1.2 millones de imágenes.

Objetivos

En este tutorial aprenderás a:

- Cargar la librería TensorFlow.js en una página Web
- Cargar un modelo pre-entrenado prefabricado de TensorFlow.js
- Utilizar el modelo MobileNet para clasificar una imagen estática

Requisitos

- Una versión reciente de Chrome o de otro navegador actualizado.
- Tu editor favorito que se **ejecute localmente en tu máquina**, por ejemplo Visual Studio Code con la extensión Live Server, **o en la Web** con servicios como codesandbox.io o glitch.com.
- Familiaridad con JavaScript.
- Familiaridad con las herramientas para desarrolladores de Chrome (DevTools) o de otro navegador.
- Cámara Web para la parte extra

Código

- Código fuente del tutorial

Tutorial

Paso 1. Esqueleto HTML

1.1. Cargar TensorFlow.js y el modelo MobileNet
Empieza creando el esqueleto de una página Web. Crea un archivo y nómbralo **index.html**. El archivo contendrá el siguiente código:

```
<!DOCTYPE html>
<html lang="en">
 <head>
   <title>Tutorial MobileNet</title>
   <meta charset="utf-8" />
   <meta http-equiv="X-UA-Compatible" content="IE=edge" />
   <meta name="viewport" content="width=device-width, initial-scale=1" />

   <!-- CSS Styling -->
   <link rel="stylesheet" href="style.css" />

 </head>
 <body>

 </body>
</html>
```

Con el atributo charset estás indicando de forma expresa la codificación de caracteres utilizada, para poder utilizar tranquilamente caracteres con tilde sin tener que recurrir a su código HTML, por ejemplo **é** para **é**.

En la etiqueta head, introduce el código para cargar la última versión de la librería TensorFlow.js y también para cargar el modelo MobileNet:

```
 <head>
  …
    <!-- TensorFlow.js library -->
   <script
     src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs@latest"
     type="text/javascript"
   ></script>

   <!-- MobileNet -->
   <script
```

```
    src="https://cdn.jsdelivr.net/npm/@tensorflow-models/mobilenet"
    ></script>
  </head>
```

Es importante incluir las dos etiquetas **script** en ese orden. Si no se incluye **tfjs** o se incluye después de **mobilenet**, tendrás un error.

En la etiqueta body, introduce el código para visualizar una imágen en la página Web:

```
  <body>
    <h1>Tutorial MobileNet</h1>
    <!-- Placeholder div for output or log messages -->
    <div id="outputMessage">Awaiting TF.js load</div>
    <img
      id="img"
      crossorigin
src="https://images.unsplash.com/photo-1562154411-dedf4b0c79bb?ixlib=rb-1.2
.1&ixid=MnwxMjA3fDB8MHxwaG90by1wYWdlfHx8fGVufDB8fHx8&auto=format&fit=crop&w
=2748&q=80"
      width="252"
      height="252"
    />
    <script src="index.js"></script>
  </body>
```

También incluirás el código para cargar el archivo **index.js**, que es el fichero donde escribirás el código JavaScript de la aplicación para la clasificación de imágenes utilizando MobileNet. Si lees la descripción del modelo MobileNet, notarás que el modelo:

> *…puede acepar como entrada cualquier elemento de imagen basado en el navegador (elementos <img>, <video>, <canvas>, por ejemplo) y devuelve una array de las predicciones más probables y sus confianzas.*

Paso 2. Clasificación de una imagen

2.1. Configurar el modelo para la inferencia en el navegador

A continuación, crea el archivo **index.js** y, luego, incluye el siguiente código para la clasificación de la imagen dentro de la función con la lógica de la aplicación, la función app():

```
async function app() {
  let model;
```

```
if (outputMessageEl) {
  outputMessageEl.innerText = "Waiting for MobileNet to load...";
}
 // Load the model.
model = await mobilenet.load();
console.log("MobileNet loaded successfully");

if (outputMessageEl) {
  outputMessageEl.innerText = "MobileNet loaded successfully";
}

// Make a prediction with the model.
const imgEl = document.querySelector("#img");
const result = await model.classify(imgEl);
console.log(result);
}
```

Si ejecutas la aplicación, en la consola verás la siguiente salida, con los mensajes de los tres console.log(): el modelo ha sido cargado con éxito y ha clasificado la imagen con una probabilidad del 71,9% como señal de tráfico. En realidad, la ejecución del modelo ha devuelto un array con todas las clasificación, y la categoría **"street sign"** es la clasificación con mayor probabilidad.

2.2. Análisis del código

En primer lugar, observa que no hay ninguna declaración **import** para utilizar el objeto **mobilenet**. El modelo pre-entrenado está disponible debido a que ha sido incluido en la página **index.html** con la etiqueta script anterior. Eso vale también para la librería TensorFlow.

Para utilizar este modelo pre-entrenado, se pone a disposición del desarrollador un API con dos métodos principales, **load** y **classify**.

1. load es un método del módulo **mobilenet** que carga el modelo, devolviendo un objeto que representa el modelo.

2. classify es un método del modelo cargado con **mobilenet.load**: acepta como parámetro un elemento HTML **img**, **canvas** o **video**, entre otros, y devuelve las primeras **topk** (parámetro opcional, 3 por defecto) posibilidades de clasificación. El

método devuelve una promesa que se resuelve en un array de clases y probabilidades, como se muestra en la consola.

El API de los modelos pre-entrados de TensorFlow utiliza JavaScript asíncrono. En este tutorial has utilizado la sintaxis async/await. Hubieses podido también utilizar la sintaxis basada en promesas, como se muestra en el código a continuación:

```
// Using the promises syntax
mobilenet.load().then((model) => {
  console.log("MobileNet loaded successfully");
  const imgEl = document.querySelector("#img");
  model.classify(imgEl).then((result) => {
    console.log(result);
  });
});
```

Ejercicios

1. Mostrar la Predicción en la página. Modificar el código para que, en lugar de mostrar la predicción solo en la consola, también se muestra en la página web dentro del elemento outputMessage.
2. Cambiar la imagen a clasificar. Cambiar la imagen por una URL distinta (ej. un perro, una flor, etc.) y observar cómo cambia la predicción de MobileNet.
3. Añadir un botón de clasificación. Añadir un botón en la página para iniciar la clasificación de la imagen, en lugar de que el modelo se ejecute automáticamente al cargar la página.