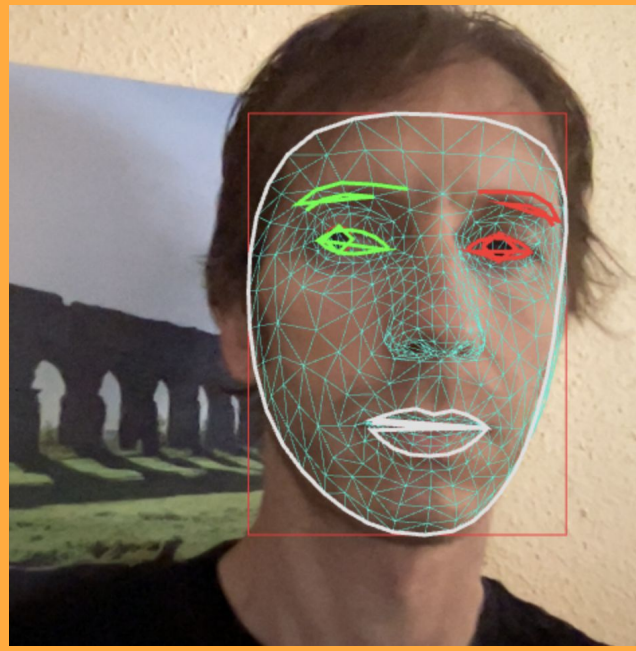
Inteligencia Ambiental

Máster Universitario en Inteligencia Artificial Aplicada



Introducción al Aprendizaje Automático en la Web con TFJS

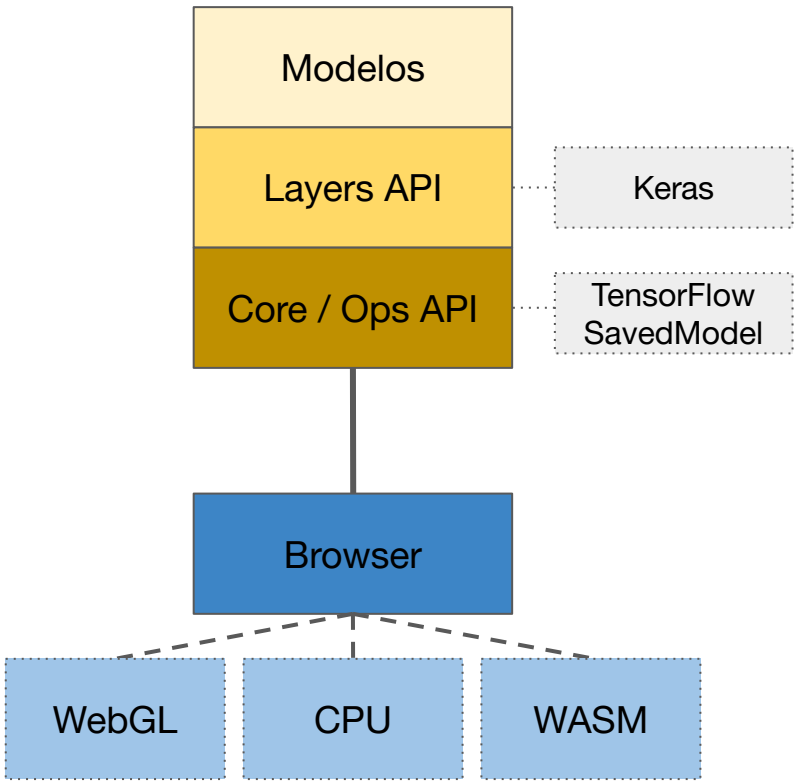
uc3m | Universidad Carlos III de Madrid

Contenido

1. API de TensorFlow.js y modelos
2. Utilizar mi primer modelo prefabricado en TensorFlow.js
3. Primera práctica

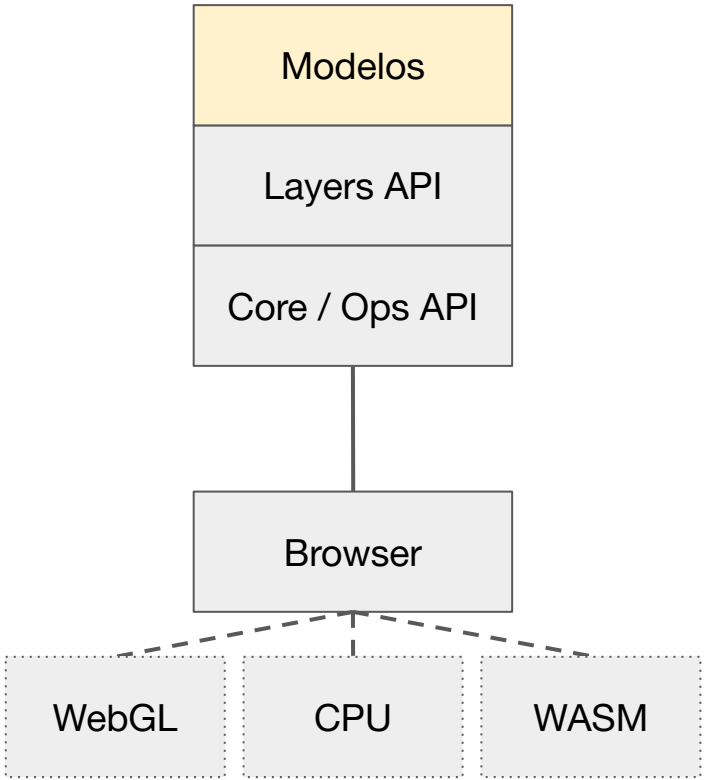
1. API de TensorFlow.js y modelos

API de TensorFlow.js



https://js.tensorflow.org/api

API de TensorFlow.js



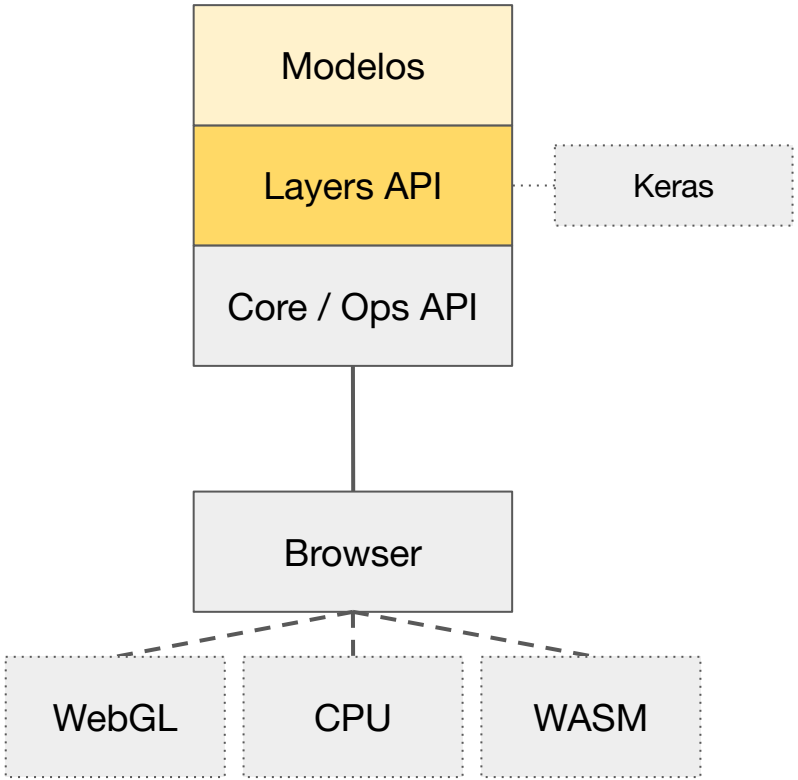
Modelos

https://www.tensorflow.org/js/models

https://github.com/tensorflow/tfjs-models

- Modelos pre-entrenados listos para usar:
- **Modelo = Datos + Algoritmo de predicción**
- Un **algoritmo** en ML lleva a cabo el reconocimiento de patrones y “aprende” de los datos: regresión lineal, árboles de decisión, redes de neuronas, KNN, …
- Un **modelo** representa lo que aprendió un algoritmo de aprendizaje automático: las reglas, los números y cualquier otra estructura de datos específica del algoritmo necesaria para hacer predicciones
- El **modelo** es el fichero que se guarda después de ejecutar un algoritmo de aprendizaje automático en los datos de entrenamiento

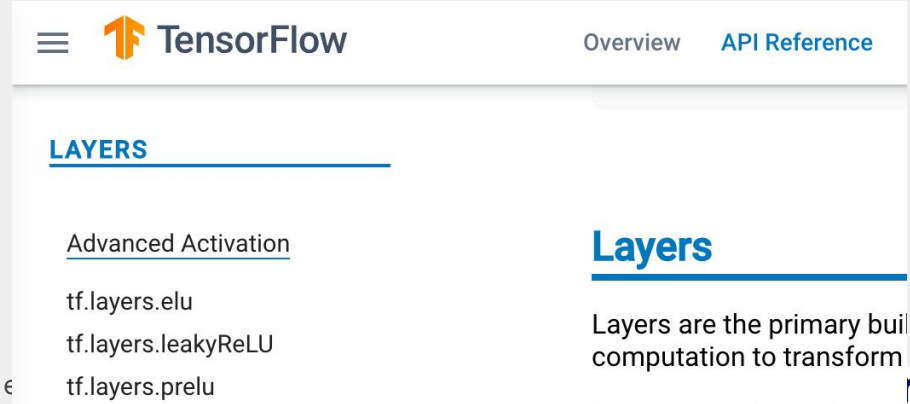
API de TensorFlow.js



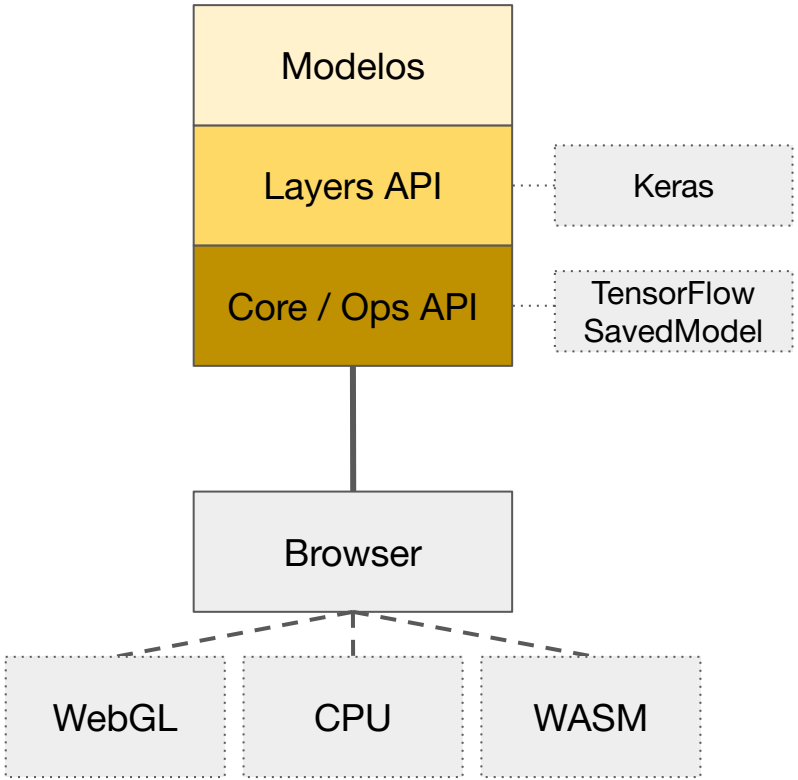
Layers API

https://js.tensorflow.org/api/latest/#Layers

- API de alto nivel
- Permite crear modelos personalizados sin tener que lidiar con las matemáticas involucradas en el aprendizaje automático
- Es análoga a la API de Keras disponible en Tensorflow (Python)
- Modelo de capas (Layers Model) en TFJS



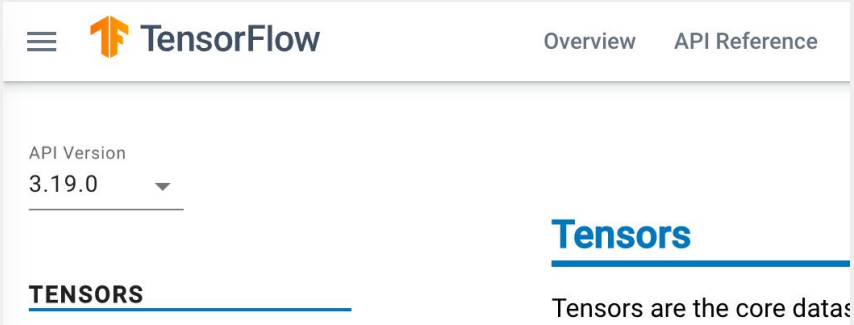
API de TensorFlow.js



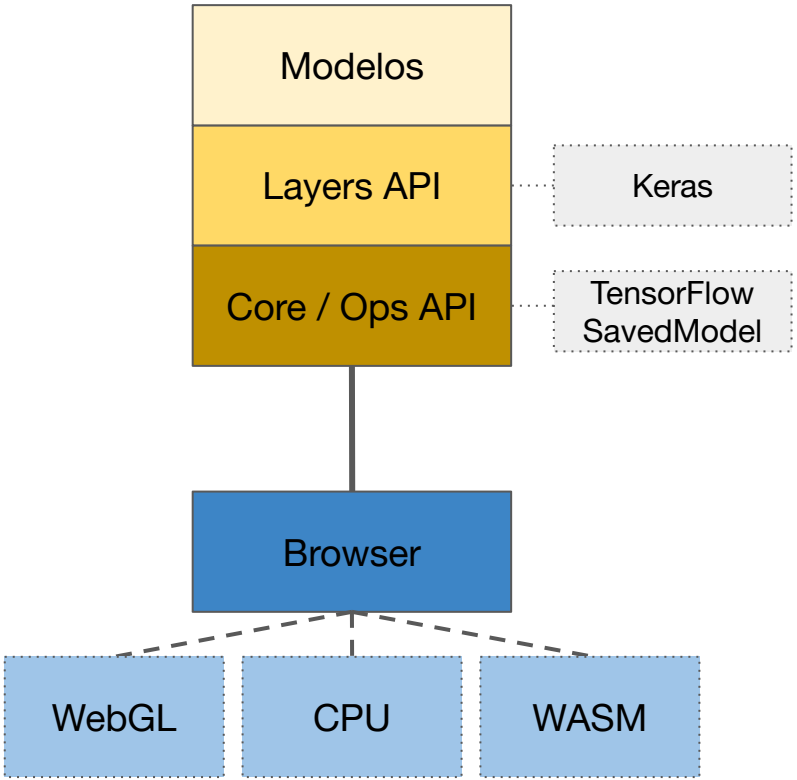
Core o Ops API

https://js.tensorflow.org/api/latest/

- API de nivel inferior
- Permite trabajar con álgebra lineal
- Modelo de grafo (Graph Model) en TFJS



API de TensorFlow.js



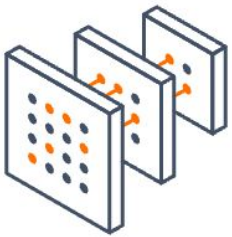
Browser y backends

- En el navegador, las API se pueden ejecutar en distintos backends y hardware
- Actualmente disponibles:

CPU	Web Assembly	WebGL
Ejecución más lenta Siempre disponible	Ejecución CPU rápida Modelos pequeños (< 10MB)	Ejecución GPU rápida Modelos grandes (>10MB)

- Nuevos estándares:
  - WebNN
  - Web GPU

Modelos de ML en TFJS



Ejecutar modelos existentes

Modelos oficiales TFJS o convertir modelos Python

Volver a entrenar modelos existentes

Aprendizaje por transferencia

Crear modelos con JavaScript

Modelos personalizados

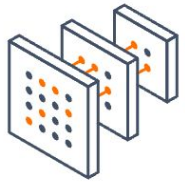
TensorFlow Lite

TensorFlow.js

TensorFlow

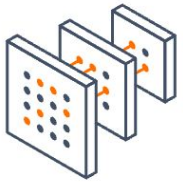
MediaPipe

Ejecutar modelos existentes



- Los **modelos pre-entrenados** ya se han entrenado en un escenario y los desarrolladores pueden reutilizarlos para casos de uso similares
- Modelos que han sido entrenados en grandes cantidades de datos y son robustos
- La documentación de algunos modelos incluye ejemplos de cómo funciona un modelo, los datos utilizados para entrenarlo y los sesgos conocidos
- El uso de modelos pre-entrenados ahorra a los desarrolladores una gran cantidad de tiempo: otros han recopilado los datos y los costos de desarrollo son menores
- Dos tipos de modelos pre-entrenados:
    - **Prefabricados**: el funcionamiento interno se oculta a través de un objeto JavaScript con métodos para cargar y ejecutar el modelo, para que sea más fácil de usar para principiantes
    - **Brutos:** es necesario escribir el código JavaScript para enviar datos al modelo y luego extraer lo datos, usando tensores y los métodos de la librería TensorFlow.js

Modelos pre-entrenados prefabricados en TFJS



Útiles para resolver tareas comunes en sistemas de Aml

https://www.tensorflow.org/js/models

Visión

- Clasificación de imágenes
- Detección de objetos
- Segmentación semántica
- Estimación de profundidad

Cuerpo humano

- Segmentación del cuerpo
- Detección de poses
- Detección facial
- Puntos de referencias faciales
- Detección de posturas de mano

Texto

- Toxicidad del texto
- Lenguaje natural para responder a preguntas
- Codificador universal de oraciones

Sonidos

- Detección de sonidos
- Reconocimiento de comandos de voz

Otros

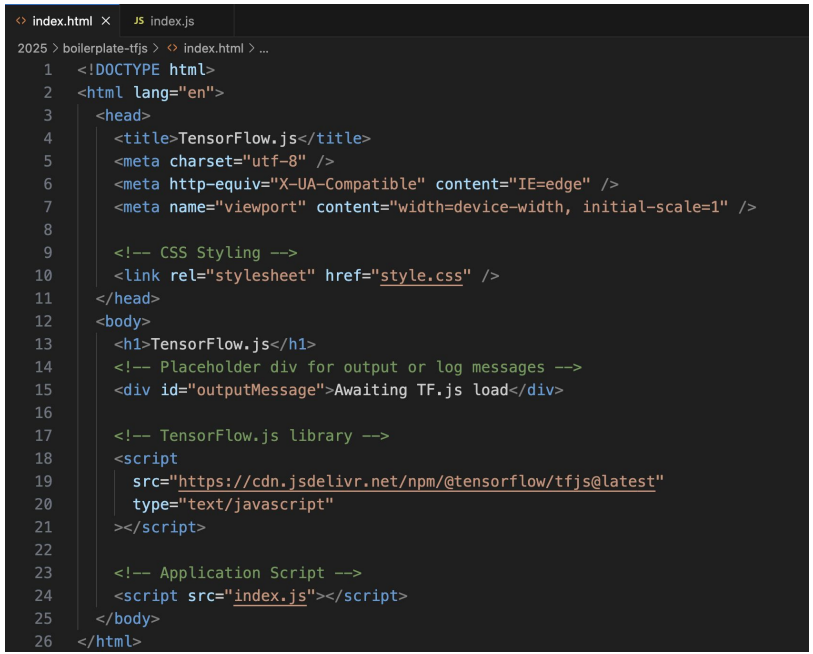
- Clasificador KNN

2. Utilizar mi primer modelo prefabricado en TensorFlow.js

Boilerplate

https://aulaglobal.uc3m.es/mod/resource/view.php?id=5252434

index.html



index.js

¿Qué es MobileNet?

- Una familia de arquitecturas de redes neuronales para la clasificación eficiente de imágenes en un dispositivo y tareas relacionadas
- Entrenando en un subconjunto de imágenes ILSVRC de ImageNet, con más de 1.2 millones de imágenes
- Dada una imágen de entrada al modelo, la **salida** es un array con las clasificaciones y su probabilidad
- 1000 clases (objetos comunes)

Tutorial 1: clasificación de imágenes con MobileNet

https://docs.google.com/document/d/19y1nwmBPok8qx90O7ZT1wHJFqG0w_y8BtJui4W63e_0

Aprender 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

¿Qué es COCO-SSD?

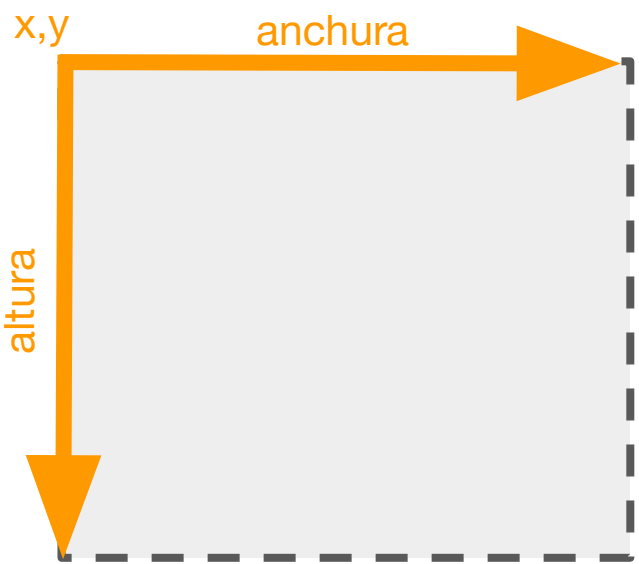
https://github.com/tensorflow/tfjs-models/tree/master/coco-ssd

- Modelo de aprendizaje automático de detección de objetos pre-entrenado, cuyo objetivo es **localizar** e **identificar** varios objetos en una sola imagen
- Proporciona las coordenadas en la imagen de un **cuadro delimitador** de objetos que ha sido entrenado para encontrar, para así proporcionar la **ubicación** de ese objeto en cualquier imagen dada
- COCO-SSD ha sido pre-entrenado para reconocer 90 objetos cotidianos comunes, como una persona, un automóvil, un gato, etc.
- El nombre tiene su origen en 2 siglas:
  - **COCO**: se refiere al hecho de que se entrenó en el conjunto de datos COCO (Common Objects in Context), que está disponible gratuitamente para que cualquiera lo descargue y lo use para entrenar sus propios modelos. El conjunto de datos contiene más de 200.000 imágenes etiquetadas que se pueden utilizar para aprender.
  - **SSD (Single Shot MultiBox Detection)**: se refiere a parte de la arquitectura que se utilizó en la implementación del modelo. No entraremos en detalle.

Cuadro delimitador

- El modelo COCO-SSD retorna una lista de objetos detectado, con la clase y su probabilidad (**class**, **score**)
- También devuelve parámetros de cuadros delimitadores de cada objeto (**bbox**)
- Los cuadros delimitadores son rectángulos que se dibujan alrededor de objetos de interés en una imagen
- Los parámetros son:
  - La coordenada x de la esquina superior izquierda de cada cuadro
  - La coordenada y de la esquina superior izquierda de cada cuadro
  - El ancho de cada caja
  - La altura de cada caja

```
[{
  bbox: [x, y, width, height],
  class: "person",
  score: 0.8380282521247864
}, {
  bbox: [x, y, width, height],
  class: "kite",
  score: 0.74644153267145157
}]
```

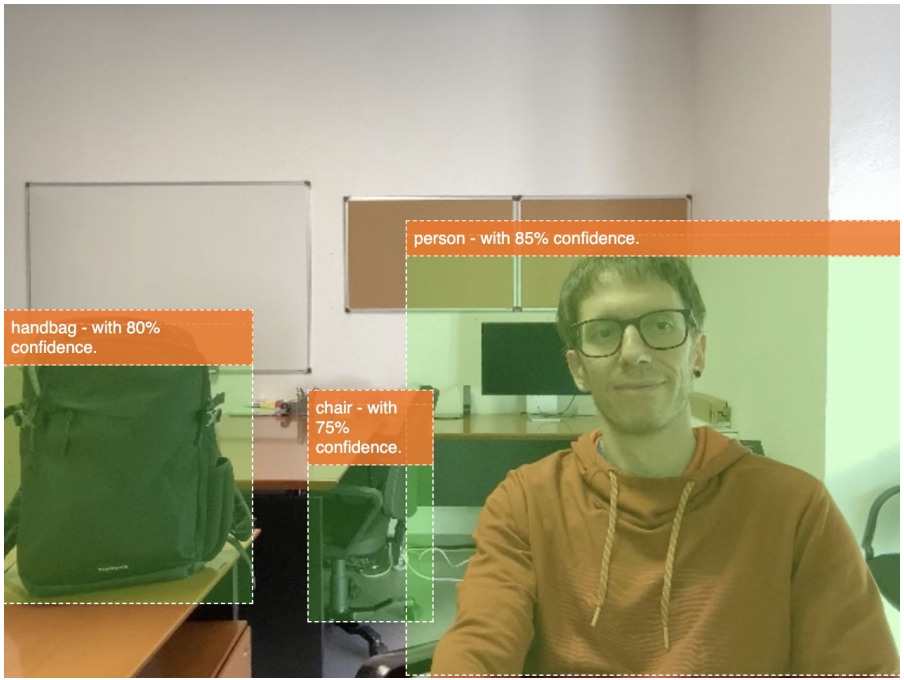


Tutorial 2: creación de una Webcam inteligente con COCO-SSD

https://docs.google.com/document/d/1Gl0g1ltZY_mbn7YgodiiXgyESc-uU_IyOezFwx8kQm8

Aprender a:

- Cargar un modelo pre-entrenado prefabricado de TensorFlow.js
- Clasificar una imagen para encontrar los cuadros delimitadores de objetos
- Utilizar las API de TFJS para recuperar fotogramas de la cámara Web como tensores
- Potenciar la cámara web utilizando los datos transmitidos desde el modelo y así resaltar los objetos encontrados



3. Primera práctica

Primera práctica (Entrega domingo 17-Nov, 23:55)

- Esta primera práctica tiene como objetivo familiarizarse con los modelos pre-entrenados de TensorFlow.js y conocer sus posibilidades y limitaciones en un contexto específico: una aplicación para la cocina inteligente.
- Utilizando el modelo COCO-SSD para la detección de objetos en tiempo real, se implementará una herramienta que pueda asistir en tareas cotidianas en la cocina, ofreciendo recordatorios y sugerencias útiles según los objetos detectados
- Enunciado de la práctica: https://docs.google.com/document/d/1GNS8ASvQzkeGCekpufwxSHwWTzCOoYUSPdpnMz85vZE

Resumen

- La librería TensorFlow.js proporciona una API JavaScript para:
  - Ejecutar modelos existentes (modelos pre-entrenados)
  - Volver a entrenar modelos existentes (aprendizaje por transferencia)
  - Crear modelos personalizados
- TFJS proporciona un conjunto de modelos pre-entrenados prefabricados para tareas comunes en sistemas AmI, por ejemplo detección de objetos, reconocimiento de comandos vocales o detección de poses
- MobileNet es una familia de arquitecturas de redes neuronales para la clasificación eficiente de imágenes en un dispositivo y tareas relacionadas
- COCO-SSD es un modelo pre-entrenado preconstruido de detección de objetos, cuyo objetivo es localizar e identificar varios objetos en una sola imagen