P1 IntAmb, Diego Collado 100405606

Detección de objetos con coco ssd

Entradas y Salidas del Modelo COCO SSD

Entradas:

- Imágenes Subidas por el Usuario: En el programa, el modelo recibe imágenes cargadas por el usuario a través de un campo de entrada en la página web “imageUpload”, y con un event listener.
- Procesamiento: TensorFlow.js maneja el procesamiento necesario, cargar el modelo con la API, llamar a sus métodos, y acceder a las predicciones para poder pintar las salidas en el canvas.

Salidas:

- Cajas Delimitadoras: Para cada objeto detectado en la imagen, el modelo proporciona las coordenadas de una caja delimitadora que son las que usamos para pintar el rectángulo.
- Clases de Objetos y Etiquetas: Junto con cada caja delimitadora, se proporciona el nombre de la clase del objeto detectado.
- Puntuaciones de Confianza: Cada detección viene con una puntuación que indica la confianza del modelo en esa detección particular.

Aplicaciones en AmI

Basándonos en el programa desarrollado, se podrían imaginar varias aplicaciones en entornos inteligentes:

- Aplicaciones de Asistencia Personal: Por ejemplo, una aplicación que ayude a las personas a identificar objetos en su entorno, útil para personas con discapacidades visuales.
- Educación y Aprendizaje: Un programa interactivo para niños que les enseñe sobre diferentes objetos y animales mediante la identificación de estos en imágenes.
- Gestión de Contenidos Digitales: Organizar automáticamente las fotografías en categorías basadas en los objetos identificados.

Errores y limitaciones

- Tras la creación del programa podemos observar que el modelo es sensible a los tamaños de las imágenes, si nuestra imagen es muy grande ocupa demasiado en la pantalla, y aunque no afecta a la funcionalidad, no es estético. Esto lo podríamos solucionar añadiendo restricciones al objeto imagen, algo muy sencillo y que para programas más funcionales tendrá que añadir.
- La segunda limitación y tal vez la más importante es el error generado o las pobres predicciones que realiza cuando la imagen es muy pequeña o los bordes no están perfectamente marcados, tras investigar en webs y artículos esto se debe a la naturaleza de la arquitectura que utiliza. En este caso podemos ver cómo detecta que donde hay un vaso con reflejos, hay un coche. La solución más sencilla es simple, solo usar imágenes de buena calidad y donde se diferencien bien los objetos, pero es improbable en un entorno real que siempre tengamos una imagen o fuente ideal. La solución más correcta sería añadir una capa de preprocesado donde consigamos tener una entrada para el modelo que sea más sencilla de analizar.

Detección de caras con MediaPipe

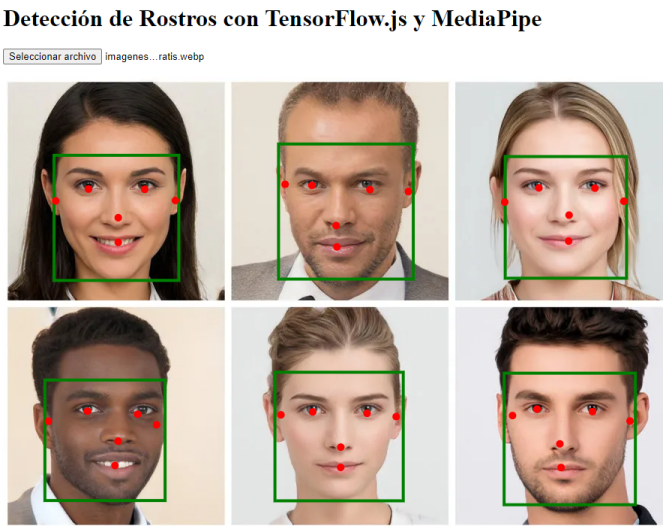
Entradas y Salidas del Modelo de det de caras MediaPipe

Entradas:

- Imágenes Subidas por el Usuario: Al igual que en el anterior programa, a través de un campo de entrada, el usuario carga imágenes que son procesadas por el modelo.
- Procesamiento: TensorFlow.js maneja la carga del modelo y la ejecución de la detección de rostros, procesando las imágenes subidas, el pipeline es el mismo pero con otro modelo precargado.

Salidas:

- Cajas Delimitadoras: El modelo proporciona las coordenadas de las cajas delimitadoras para cada rostro detectado igual que coco ssd.
- Puntos Clave del Rostro: Además de las cajas, las predicciones nos dan puntos clave en el rostro, como los ojos, la nariz y la boca que también pintaremos en el canvas



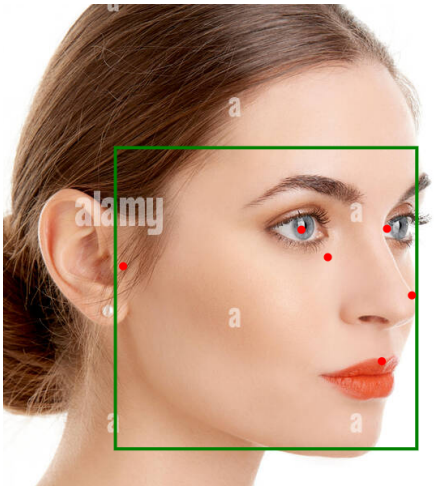
Aplicaciones en AmI

- Sistemas de Seguridad y Control de Acceso: Utilizar la detección de rostros para sistemas de seguridad residencial o control de acceso en edificios.

- Interfaz de Usuario Mejorada: Integrar la detección de rostros en aplicaciones para personalizar la interacción del usuario basada en su identificación o expresiones faciales.
- Asistencia Personalizada: Por ejemplo, dispositivos que ajustan su funcionalidad o contenido según el usuario identificado por la detección de rostros

Errores y limitaciones

- Dificultad con Rostros Parcialmente Girados: He notado que el modelo puede tener dificultades para detectar rostros que no están orientados frontalmente o que están parcialmente girados.
- Sensibilidad a la Calidad de la Imagen: El modelo puede fallar o ser menos preciso con imágenes de baja calidad, pequeñas o con rostros borrosos.
- Limitación de MediaPipe: Aunque MediaPipe es eficiente y rápido, puede tener limitaciones en cuanto a la variedad y complejidad de los rostros que puede detectar con precisión, especialmente en condiciones de iluminación pobres o fondos complejos.



Posibles soluciones

- Mejoras en el Preprocesamiento de Imágenes: Implementar técnicas avanzadas de preprocesamiento para mejorar la calidad de las imágenes antes de la detección, como ajustes de iluminación o filtrado de ruido.
- Combinar con Otros Modelos: En casos donde la orientación del rostro es un problema, combinar MediaPipe con otros modelos que sean robustos a diferentes orientaciones y expresiones faciales.

Prueba Extra

Tal y como vimos en clase añadir un objeto para detectar por la webcam no era muy complicado, los resultados son igual de buenos que con las imágenes, y de la misma manera las limitaciones también son las mismas