Facultad de Ingeniería

Visión computacional

Cesar Augusto Puente Montejano

Brandon Torres Barajas

08/12/21

Tercer examen parcial

a) Planteamiento del problema a resolver.

Implementar un detector de objetos utilizando el módulo DNN de OpenCV.

1.Investigar sobre el dataset de Imágenes de objetos MS COCO. Decir quién y cuándo lo creó. Qué contiene, cuantas clases y número de imágenes. Agregar para que tipo de problemas se ha utilizado y el enlace de donde se puede bajar públicamente.

2.Utilizar la biblioteca OpenCV para implementar un programa que detecte objetos que pertenezcan a alguna(s) de clase(s) del dataset MS COCO.

3.Para probar su implementación, debe tomar una foto (o video, si su capacidad de cómputo lo permite) usted mismo donde aparezcan objetos de las clases de MS COCO. Si es muy difícil hacer una foto o video con las clases que usted eligió para probar, puede utilizar alguna imagen o video del dominio público y poner la referencia de donde lo obtuvo. El caso es que NO utilice fotos o videos del mismo dataset MS COCO.

b) Descripción de la solución.

El MS COCO es un conjunto de datos de detección, segmentación y consulta de objetos a gran escala publicado por Microsoft. Los estudiantes de aprendizaje automático y visión por computadora utilizan popularmente el conjunto de datos COCO para la resolución de varios proyectos que involucren la visión computacional.

Con COCO, Microsoft presentó un conjunto de datos visuales que contiene una gran cantidad de fotos que representan objetos comunes en escenas cotidianas complejas. Esto distingue a COCO de otros conjuntos de datos de reconocimiento de objetos que pueden ser sectores sumamente específicos de inteligencia artificial. Dichos sectores, o problemas, incluyen la clasificación de imágenes, la localización del cuadro delimitador de objetos o la segmentación a nivel de píxel.

Las clases de conjuntos de datos COCO para la detección y el seguimiento de objetos incluyen en total 80 objetos entrenados previamente.

```
'person', 'bicycle', 'car', 'motorcycle', 'airplane', 'bus', 'train', 'truck', 'boat',
'traffic light', 'fire hydrant', 'stop sign', 'parking meter', 'bench', 'bird', 'cat',
'dog', 'horse', 'sheep', 'cow', 'elephant', 'bear', 'zebra', 'giraffe', 'backpack',
'umbrella', 'handbag', 'tie', 'suitcase', 'frisbee', 'skis','snowboard', 'sports
ball', 'kite', 'baseball bat', 'baseball glove', 'skateboard', 'surfboard', 'tennis
racket', 'bottle', 'wine glass', 'cup', 'fork', 'knife', 'spoon', 'bowl', 'banana',
'apple', 'sandwich', 'orange', 'broccoli', 'carrot', 'hot dog', 'pizza', 'donut',
'cake', 'chair', 'couch', 'potted plant', 'bed', 'dining table', 'toilet', 'tv',
'laptop', 'mouse', 'remote', 'keyboard', 'cell phone', 'microwave', 'oven', 'toaster',
'sink', 'refrigerator', 'book', 'clock', 'vase', 'scissors', 'teddy bear', 'hair
drier', 'toothbrush'
```

Este dataset contiene en total 330.000 imágenes, de las cuales 200.000 ya cuentan con una etiqueta definida, lo que mejora los resultados de la clasificación. Se puede descargar directamente de su página oficial, dependiendo de lo que cada usuario necesite, desde https://cocodataset.org/#download.

Pero para que esto funcione se tiene que utilizar en conjunto una red neuronal profunda (DNN). En este caso se utilizó TensorFlow debido a que se obtuvieron mejores resultados que con la DNN Coffee. TensorFlow es la plataforma de aprendizaje profundo más importante del mundo. Este

desarrollo open-source de Google va más allá de la Inteligencia Artificial debido a su flexibilidad y su gran comunidad de desarrolladores, lo que lo ha posicionado como la herramienta líder en el sector del Deep Learning. Es el sistema de aprendizaje automático de segunda generación de Google Brain, liberado como software de código abierto en 9 de noviembre del 2015.

TensorFlow es una gran plataforma para construir y entrenar redes neuronales, que permiten detectar y descifrar patrones y correlaciones, análogos al aprendizaje y razonamiento usados por los humanos. Una de las aplicaciones más interesantes está en los teléfonos. Por ejemplo, el Pixel 2 incluye efecto bokeh con una sola cámara. Se crea un modo retrato que separa a la persona del fondo, cuando esto era algo reservado para dispositivos con doble cámara. Y esto se consigue con el TensorFlow de Machine Learning, entrenando un modelo de TensorFlow en el backend, pero también ejecutándolo en el propio teléfono. No es tarea sencilla. La rapidez de la solución y el fantástico resultado que tiene es todo un hito tecnológico. Google ha sido capaz de imitar un efecto propio de la física óptica con sólo software y aprendizaje profundo.

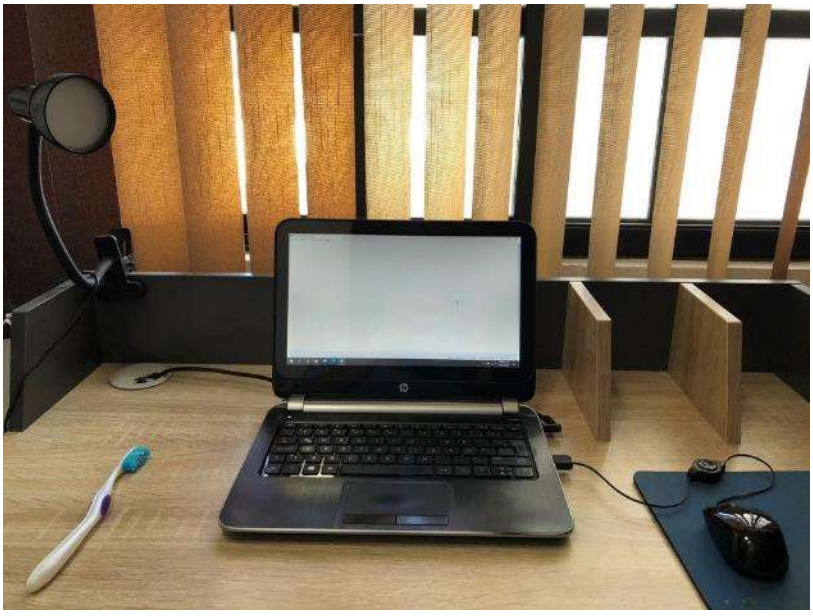
Para la implementación de la solución del examen, lo primero que se hace es cargar el MSCOCO, con el archivo indicado y solo de lectura, y la DNN. Este conjunto de archivos se encuentra dentro de la carpeta “archivos”.

```
4  #cargar las clases de MSCOCO
5  with open('archivos/object_detection_classes_coco.txt', 'r') as f:
6      class_names = f.read().split('\n')
```

La red neuronal profunda se carga dentro de la variable “model”, donde se le indica el modelo a utilizar en el ejercicio, el archivo de configuración y el framework. En este último parámetro es donde se le indica que será la DNN TensorFlow.

```
11 #cargar el DNN
12 model = cv2.dnn.readNet(model='archivos/frozen_inference_graph.pb',
13                         config='archivos/ssd_mobilenet_v2_coco_2018_03_29.pbtxt.txt',
14                         framework='TensorFlow')
```

Después se lee la imagen con los objetos que quieren ser detectados. En este caso, ara comprobar el correcto funcionamiento del detector, se utilizaron las siguientes imágenes:



```
16 #leer imagen
17 image = cv2.imread('foto1.jpeg')
18 #image = cv2.imread('foto2.jpeg')
```

Después, se crea una imagen blob, que es la imagen cargada, pero con algunos filtros. Image es la imagen fuente, size cambia el tamaño de la imagen, mean modifica los valores BGR de la imagen y swapRB cambia los canales R y B de la imagen, convirtiéndola en formato RGB.

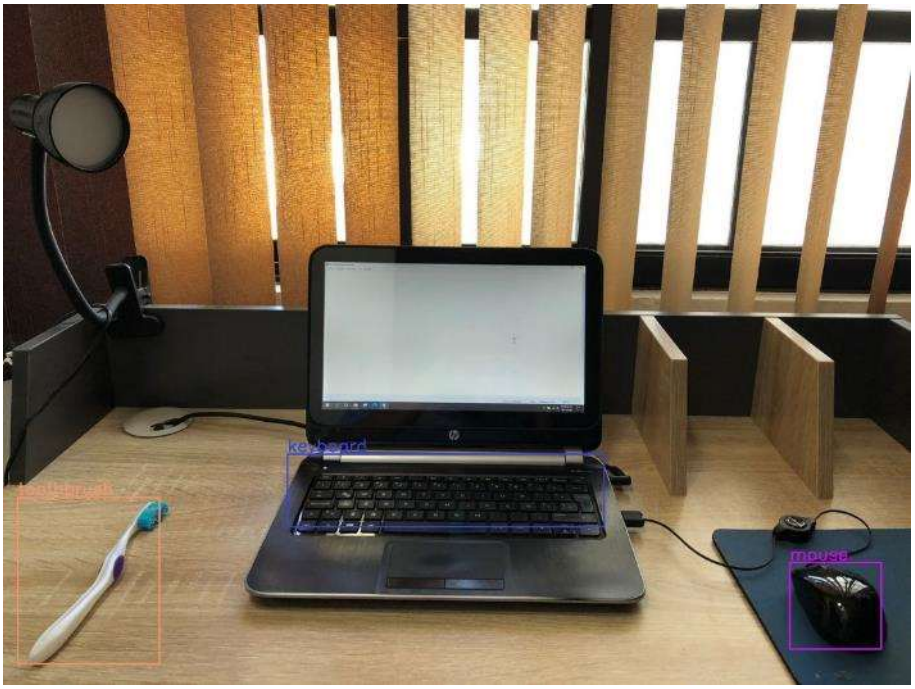
```
20
21 blob = cv2.dnn.blobFromImage(image=image, size=(700, 700), mean=(25, 25, 25), swapRB=True)
22
23 model.setInput(blob)
24
25 output = model.forward()
```

Después de esto, se carga la imagen blob al modelo y se ejecuta para obtener los objetos detectados. A cada objeto detectado se le dibuja un rectángulo que lo delimita y se le agrega el nombre de su clase.

c) Descripción de los resultados.

Los resultados obtenidos con la DNN TensorFlow fueron los que más se acercaron a lo esperado, en cuanto a cantidad de objetos detectados. Se hizo una prueba utilizando Coffee como DNN, pero la implementación no logró que se detectaran correctamente los objetos y se descartó su uso en el programa.

Para el primer caso, se esperaba que se detectaran tres objetos, de acuerdo con la lista de objetos que el conjunto de la base tiene.



Se pudieron detectar los objetos toothbrush, keyboard y mouse, cada uno delimitado decentemente con un rectángulo y de color diferente. Al momento de realizar el algoritmo, a veces se detectaba solamente la laptop, dejando de lado los otros dos objetos. Esto cambia si los parámetros de la función con la que obtenemos la imagen blob son modificados.

La imagen de arriba muestra los mejores resultados que se obtuvieron en esta escena. En un principio se esperaba que se detectara un objeto parecido a una lámpara, pero parece ser que en el dataset no existe esa clase.

Para el segundo caso, los parámetros al obtener la imagen blob fueron modificados nuevamente con tal de obtener mejores resultados para esta escena, logrando que se detecten los objetos deseados, que esta vez fueron solamente dos.

Los objetos que se detectaron pertenecen a las clases person y car, cada uno delimitado, nuevamente.

d) Discusión.

Los resultados demostraron que las variaciones de los parámetros son de vital importancia, al menos al tratar de reconocer objetos en determinados ambientes. Hay ocasiones en las que de un valor entero a otro que es consecutivo, por ejemplo 4 y 5, lo que se obtiene es completamente diferente a lo que se espera. Este fue el caso de la primera imagen, los objetos a veces eran etiquetados incorrectamente o no eran detectados, y todo era por la variación de los parámetros que utilizan las funciones. En el segundo caso no fue tan marcado esta realidad, pero si llegó a pasar que los objetos eran delimitados de forma extraña y a veces no eran considerados resultados satisfactorios.

Creo que este tipo de reconocimientos de imágenes es muy útil y práctico, pero se debe tener en cuenta que hay que hacer cambios para que funcione en diferentes entornos entregando los mejores resultados que se creen posibles, inconveniente que creo que ya ha sido más que resulto en el día a día con sistemas mucho más sofisticados.

e) Conclusión.

Nuevamente pude comprobar la facilidad con la que OpenCV ayuda a todas las personas interesadas en conocer sobre la visión computacional al momento de implementar código especializado en el área. En esta ocasión fue un poco más elevada la complejidad, debido a que al ser un detector de objetos lo que se pedía, este tiene que ser capaz de identificar y limitar las partes correspondientes de cada una de las imágenes localizadas, que esta vez funcionan como objetos.

Me queda claro que el clasificar y detectar objetos son solo los primeros pasos si se quiere llegar a trabajar con estos recortes y realizar un sistema mucho más específico y elaborado. En este ejercicio solamente se localizaban las imágenes y se les daba su correspondiente nombre de clase, pero los parámetros de las funciones tienen que estar bien definidos porque al no aceptar tanta libertad, a veces los resultados varían mucho entre un número y otro por variable. Sin embargo, creo que sigue siendo una forma muy fácil y accesible de trabajar con la visión computacional.

Fuentes

https://viso.ai/computer-vision/coco-dataset/

https://puentesdigitales.com/2018/02/14/todo-lo-que-necesitas-saber-sobre-tensorflow-la-plataforma-para-inteligencia-artificial-de-google/