Detección de objetos en OpenCV

Planteamiento del problema

Implementar un detector de objetos utilizando el modulo DNN de OpenCV, este puede usar cualquier DNN y framework soportados por OpenCV, pero para obtener un buen resultado esta debe de ser pre-entrenada con el dataset MS COCO.

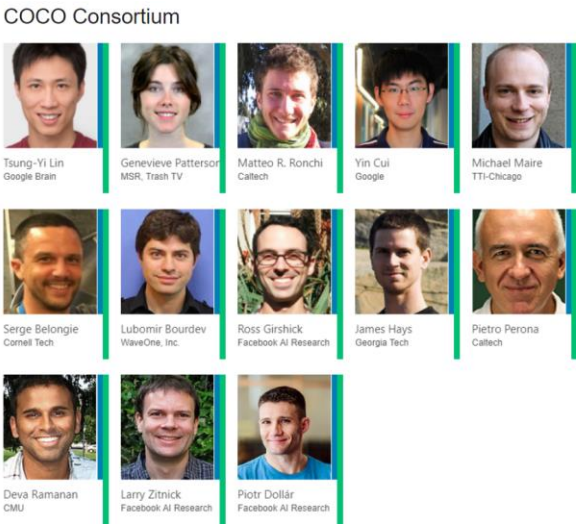
Descripción de la solución

MS COCO

Microsoft Coco es un dataset que consiste en 328 mil imágenes que funcionan para la detección de objetos, segmentación, obtener puntos clave. MS COCO tiene al menos 80 clases de objetos, desde persona, carro, hasta un cepillo de dientes, estas clases son de objetos del diario, se usa un archivo de texto donde se tienen todas las clases presentes en el MS COCO para la detección de objetos.

La primera versión de MS COCO fue lanzada en 2014, esta contenía 164 mil imágenes, divididas en 83 mil de entrenamiento, 41 mil de prueba y 41 mil de validación. Luego en 2017 se incremento la cantidad de imágenes a 123 mil.

Este fue creado por un grupo de desarrolladores que trabajan para diferentes empresas y universidades como Google, Facebook, TTI-chicago, Calthech y WaveOne.



Se pueden descargar los archivos necesarios para trabajar con el dataset en la liga https://cocodataset.org/#download. O usar directamente la api en https://github.com/cocodataset/cocoapi

Yuxin Wang, Weibin Liu, Weiwei Xing usaron el dataset MS COCO para resolver un problema de detección de objetos en la que esta era demasiado lenta con la red Banlanced-RetinaNet, la cual tienes tres niveles, primero la extracción de estados, luego la regresión de objetos y finalmente la clasificacionde objetos, pero al momento de utilizar MS COCO el rendimiento aumento considerablemente.

DNN MobileNeT

Para la resolución del problema de detectar objetos con MS COCO se uso el modelo DNN para la detección de objetos y previamente entrenado con MS COCO de nombre MobileNet, esta usa circunvoluciones separables en profundidad. Reduce significativamente el número de parámetros en comparación con la red con convoluciones regulares con la misma profundidad en las redes. Esto da como resultado redes neuronales profundas ligeras.

Una convolución separable en profundidad se realiza a partir de dos operaciones:

- *Depthwise Convolution*
- *Pointwise Convolution*

Esta fue desarrollada por Google de tal manera que nos da un excelente punto de partida para entrenar a nuestros clasificadores.

Código para la solución

```
...
1  #Importacion de las librerias
2  import cv2 as cv
```

Cargar OpenCV

```
4   #abrir el archivo de clases
5   with open('object_detection_classes_coco.txt', 'r') as f:
6       class_names = f.read().splitlines()
7
```

Obtener el archivo de las clases, a la variable se le asigna su contenido

```
#cargar el modelo DNN, en este caso se uso mobilnet
model = cv.dnn.readNet(model='frozen_inference_graph.pb',
                       config='ssd_mobilenet_v2_coco_2018_03_29.pbtxt.txt',
                       framework='TensorFlow')      You, now • Uncommitted chan
```

Cargar el modelo DNN mobileNet, con la red neuronal

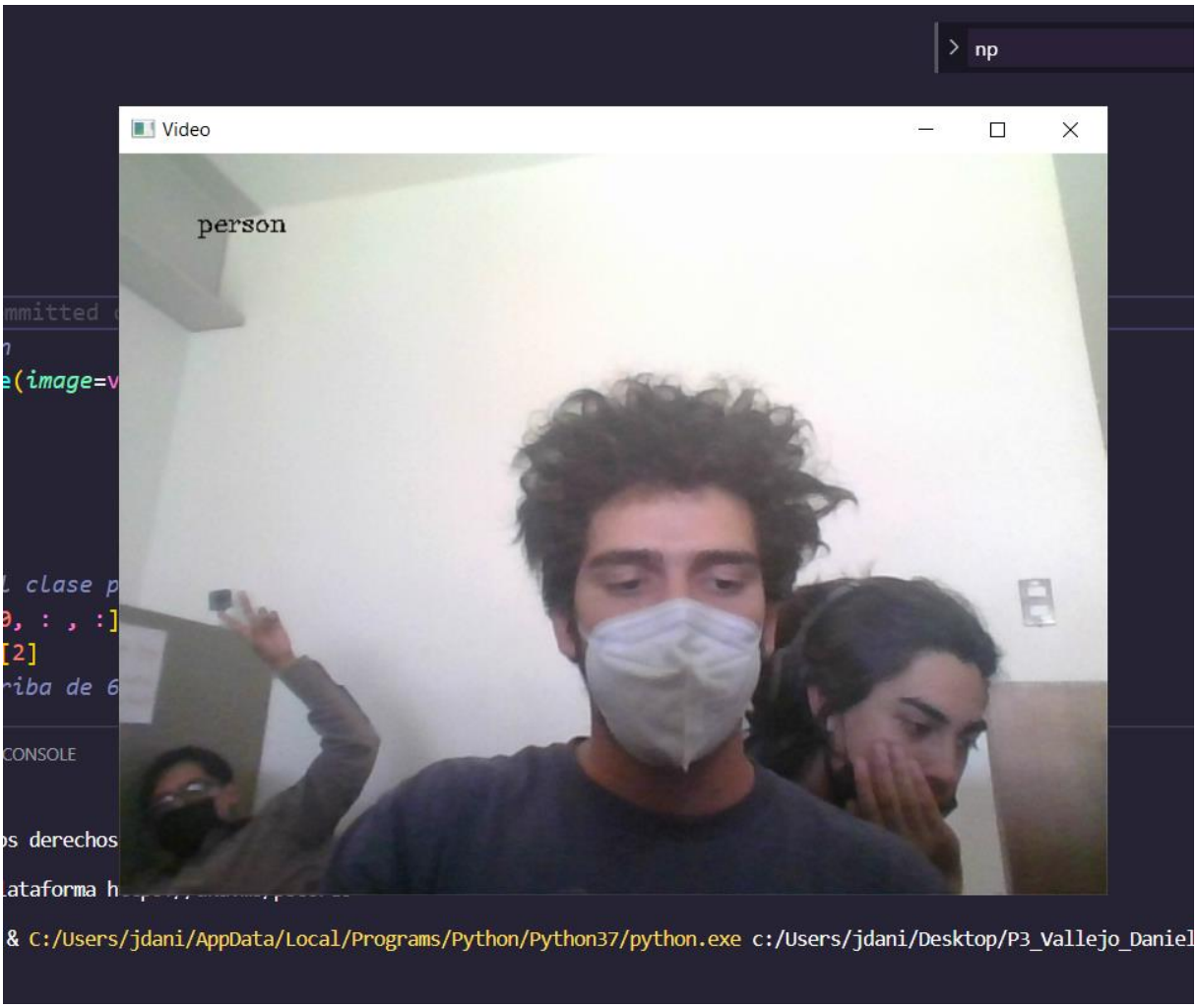
```
while 1:
    #leer la camara
    sucess, video = cap.read()

    #crear un blob de la imagen
    blob = cv.dnn.blobFromImage(image=video, size=(300, 300), mean=(104, 117, 123), swapRB=True)

    #aplicar el modelo al blob
    model.setInput(blob)
    output = model.forward()

    #ciclo para detectar a cual clase pertenece la imagen
    for detection in output[0,0, : , :]:
        confidence = detection[2]
        #si la confianza es arriba de 6 dibujar el nombre de la clase a la que pertenece
        if confidence > .6:
            class_id = detection[1]
            class_name = class_names[int(class_id)-1]
            cv.putText(video, class_name, (50,50), cv.FONT_HERSHEY_COMPLEX, 0.5, (0,0,0))

        #mostrar la imagen      You, 1 second ago • Uncommitted changes
        cv.imshow("Video", video)
        if cv.waitKey(1) == ord('q'):
                break
cv.destroyAllWindows()
```

En un ciclo infinito se lee la cámara, luego se crea un blob con los tamaños necesarios para poder procesar la imagen a través del modelo y luego se inserta el blob en el modelo y en un ciclo se detecta a que clase pertenece la imagen, si la confianza es arriba del %60 se muestra la clase en la esquina superior derecha de la ventana en la que aparece la imagen obtenida por la cámara del equipo.

Descripción de los resultados

En este caso yo use el video obtenido por la cámara de mi equipo y se puede obtener un desempeño satisfactorio, pero no el mejor, en ciertos momentos se atora y se retrasa con la imagen en vivo, sin embargo esto es raro.

Captura que muestra el funcionamiento correcto del programa:



Discusión

Detecta correctamente la imagen y el fondo, por lo que el uso de MobileNet y MS COCO es una buena opción para la detección de objetos, sin embargo para hacerlo en tiempo real es necesario un equipo más robusto o sea mayor velocidad de cómputo, en mi caso solo estoy usando el CPU un inter Core i5 de séptima generación, pero en general su desempeño es correcto.

Conclusión

El uso de OpenCV permite un gran procesamiento de imágenes que se pueden usar para resolver muchos tipos de problemáticas, y la detección de objetos es una de ellas, además esto es sin necesidad de patente o de pagar alguna licencia por lo que actualmente es de muy fácil acceso.

Referencias

Se uso parte del código que esta publicado en la página: https://learnopencv.com/deep-learning-with-opencvs-dnn-module-a-definitive-guide/ y se adapto para el uso de la cámara en tiempo real.

https://www.spiedigitallibrary.org/journals/journal-of-electronic-imaging/volume-30/issue-3/033009/Balanced-RetinaNet-solving-the-imbalanced-problems-in-object-detection/10.1117/1.JEI.30.3.033009.full?tab=ArticleLinkFigureTable